НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ О.В. Коваль
(підпис)

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Система моніторингу електронної черги вступників КПІ»

Виконав: студент IV курсу, групи ТМ-61

Гажієнко Алла Сергіївна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник старший викладач, Мірошниченко І.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
(підпис)

Рецензент старший викладач, к.т.н., Воробйов М.В _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Гажієнко Алла Сергіївна _____
(прізвище, ім’я, по батькові)

1. Тема роботи _”Система моніторингу електронної черги вступників КПІ”
керівник роботи Мірошниченко І.В., ст.викладач _____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _Мова програмування C#, платформа Microsoft Visual Studio 2019, база даних MS SQL

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _аналіз існуючих алгоритмів систем моніторингу, вибір методів та інструментів для розробки системи , розробка архітектури системи, вибір та реалізація засобів розробки програмного забезпечення, розробка додатків управління системи

5. Перелік ілюстративного матеріалу
Пропонована система реєстрації абітурієнтів, логічна модель бази данних, приклади роботи додатків, перспективи розвитку проектів

6. Дата видачі завдання ”_10__”_вересня___2019___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.09.2019-30.09.2019	
2.	Вивчення та аналіз задачі, пошук літератури	01.10.2019-30.12.2019	
3.	Розробка архітектури та загальної структури системи	18.01.2020-14.02.2020	
4.	Розробка структур окремих підсистем	15.02.2020-28.02.2020	
5.	Програмна реалізація системи	29.02.2020-27.03.2020	
6.	Оформлення пояснювальної записки	01.05.2020-17.05.2020	
7.	Захист програмного продукту	17.05.2020-29.05.2020	
8.	Передзахист	10.06.2020	
9.	Захист	17.06.2020	

Студент _____ _Гажієнко А.С._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Мірошниченко І.В._____
(підпис) (прізвище та ініціали,)

ЗМІСТ

АНОТАЦІЯ

У роботі розглядаються питання розробки систем моніторингу електронної черги вступників КПІ. Мета роботи - дослідити проблеми, пов’язані з автоматизацією процесів реєстрації абітурієнтів і з використанням технології використання протоколу TCP для взаємодії додатків, які гарантують відправку повідомлень і широко використовується в різних сучасних на сьогодні програм. Дана технологія реалізована в проекті на основі класів System.Net.Sockets.Socket (класи TcpClient і TcpListener).

В процесі роботи досліджувалися програмні продукти і технології, які використовуються для створення систем взаємодії додатків з базами даних MS SQL на основі класів System.Data.SqlClient, який є постачальником даних .NET FrameWork для джерел даних MS SQL Server.

Результат роботи є програмний комплекс «Система моніторингу електронної черги вступників КПІ», який складається з 5 додатків і бази даних MS SQL, опис процесу створення систему моніторингу електронної черги вступників КПІ, опис розгортання і установки програмного комплексу, керівництва користувача і програміста.

Розробка систему здійснюється в середовищі Microsoft Visual Studio 2019, SQL Server Management.

Об’єм роботи 73 сторінок, 50 ілюстрацій, 2 таблиці, 17 використаних джерел, 2 додатки.

Ключові слова: СИСТЕМА МОНІТОРИНГУ ЕЛЕКТРОННОЇ ЧЕРГИ ВСТУПНИКІВ КПІ, MICROSOFT SQL SERVER, TCPCLIENT, C#, VISUAL STUDIO.

ABSTRACT

In work was looking the questions of developing systems for monitoring the electronic queue of KPI. The purpose of the work is to investigate the problems associated with the automation of registration processes for applicants and using technology using the TCP protocol for application interaction, which guarantees message delivery and is widely used in various programs that exist today. This technology was implemented in a project based on the System.Net.Sockets.Socket class (TcpClient and TcpListener classes).

In the process, we studied software products and technologies that are used to create application interaction systems with MS SQL databases based on the System.Data.SqlClient class, which is the .NET FrameWork data provider for MS SQL Server data sources.

The result of the work is the software package “The system of monitoring electronic cherries for entering KPI”, consisting of 5 applications and the MS SQL database, a description of the process of building the system for monitoring electronic cherries for entering KPI, a description of the deployment and installation of the software package, user and programmer manuals.

The development was carried out in Microsoft Visual Studio 2013, SQL Server Management Studio, AIIFusion ERwin Data Modeler by CA.

The volume of work is 73 pages, 50 illustrations, 2 tables, 17 sources used, 2 applications.

Keywords: ELECTRONIC MONITORING SYSTEM OF THE INTERIOR KPI, MICROSOFT SQL SERVER, TCPCLIENT, C #, VISUAL STUDIO

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПС Програмна система

SDK Software Development Kit — набір із засобів розробки

КПІ Київський політехнічний інститут

SQL Structured query language — Мова структурованих запитів

LINQ Language Integrated Query — запити, інтегровані в мову

TCP Transmission Control Protocol — протокол керування передачею

XML Extensible Markup Language — Розширювана мова розмітки

СУБД Система управління базами даними

ПЗ Програмне забезпечення

ВСТУП

Електронна черга абітурієнтів — це програмно апаратний комплекс, який дозволяє формалізувати і оптимізувати управління потоками абітурієнтів. Головна мета системи електронної черги — цілеспрямований напрямок абітурієнтів всередині приймальної комісії, інформації про найбільш затребувані спеціальностях і т.д.

Використання електронних системи управління потоками абітурієнтів в університетах допомагають змінити і підвищити якість обслуговування. У випадку необхідності організовують запис абітурієнтів на прийом по часу і даті. Системи електронних черг дозволяють на основі в процесі роботи даних оптимізувати обслуговування або розробляти нові методики, а також оперативно їх обробляти. Наслідком застосування такої системи є поліпшення загального клімату обслуговування і більш високий коефіцієнт роботи персоналу приймальних комісій.

Система електронної черги відрізняється від систем «виклику клієнта» тим, що дозволяє ввести гнучко налаштований алгоритм управління потоком абітурієнтів, вести облік і статистику роботи секретарів приймальної комісії і інтенсивності потоків, що дозволяє ефективно планувати навантаження на працівників університету, а також використовувати інформаційне табло для відображення поточної черги.

Розробка системи моніторингу електронної черги вступників КПІ і присвячена дана робота.

1 ПОСТАНОВКА ЗАДАЧІ

Необхідно розробити систему моніторингу електронної черги вступників КПІ.

Основні функції розробленої системи:

- черга повинна бути одна для однієї людини;
- система повинна автоматично виводити на загальнодоступному моніторі номер викликаного абітурієнта;
- секретарі, які обробляють заявки, повинні бути забезпечені зручною системою відстеження стану черги і виклику абітурієнта для обслуговування;
- повинен бути доступний інтерфейс відображення поточної черги на загальнодоступному моніторі;
- реалізувати можливість закріплення за робочим місцем секретаря пріоритетних і не пріоритетних спеціальностей.

В рамках розробленої системи повинна бути реалізована клієнт-серверна конфігурація, в рамках якої повинна бути спроектована і створена база для збереження даних на основі Microsoft SQL SERVER і реалізовані наступні додатки:

- сервер для обробки запитів клієнтських додатків до бази даних і відправлення їм відповідей на основі мережевої взаємодії по протоколу TCP;
- додаток для виводу інформації на великий екран (за допомогою проектора) у тому місці, де будуть чекати абітурієнти;
- додаток для секретарів, що дозволяє визивати абітурієнтів;
- додаток для менеджера, який реєструє абітурієнтів в системі;
- додаток для адміністратора, який зміг би конфігурувати параметри столів і відслідковувати черги абітурієнтів.

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

КПІ ім. Ігоря Сікорського є одним з найбільших навчальних закладів Європи. У ньому навчається майже 25 тисяч студентів, аспірантів і докторантів, у тому числі й студенти-іноземці з країн близького та далекого зарубіжжя.

В університеті працюють 18 факультетів, 9 навчально-наукових інститутів, декілька науково-дослідних інститутів і наукових центрів. Здійснюється підготовка бакалаврів, спеціалістів та магістрів, кандидатів і докторів наук. Університет має власне видавництво "Політехніка". Серед викладачів КПІ – більше 500 професорів і понад 1300 доцентів.

Аудиторії та лабораторії оснащені сучасним обладнанням, упроваджуються новітні технології навчання з використанням комп'ютерних мереж. Усе це дозволяє забезпечити якість освіти, рівень якої відповідає стандартам кращих закордонних університетів.

Будівлі інститутів і факультетів КПІ ім. Ігоря Сікорського та університетські гуртожитки розкинулися на площі близько 120 гектарів. Це справжнє місто в місті. Університет має власний Центр культури та мистецтв, сучасний спортивний комплекс, поліклініку, чотири спортивно-оздоровчі бази на Дніпрі, Чорному морі та в Карпатах.

КПІ ім. Ігоря Сікорського є базовою організацією Державної інформаційної мережі вищих навчальних закладів і інститутів Національної академії наук URAN, яка приєднана до Європейської освітньої мережі GEANT.

КПІ ім. Ігоря Сікорського співпрацює з технічними університетами десятків країн світу, багатьма міжнародними організаціями (UNESCO, UNIDO, WIPO, NATO, EDNES, ICSU, CODATA та ін.) та найвідомішими корпораціями й фірмами (MOTOROLA, SIEMENS, SAMSUNG, INTEL тощо), бере участь у міжнародних освітніх, наукових проектах і програмах.[2]

2.1 Структура факультетів КПІ

При реєстрації абітурієнт повинен обрати спеціальність, на яку буде вступати, тому перш ніж переходити до проектування ПЗ необхідно визначити структуру факультетів, спеціальностей і спеціалізацій КПІ.

NameFaculty	SpecialtyCode	NameSpecialty	Specialization
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ	124	Системний аналіз ДП	• Системний аналіз і управління • Системний аналіз фінансового ринку
	122	Комп'ютерні науки ІТ	• Системи штучного інтелекту • Інтелектуальний аналіз даних в управлінні проектами • Інформаційні системи і технології проектування • Системне проектування сервісів
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ	172	Телекомунікації та радіотехніка	
ВИДАВНИЧО-ПОЛІГРАФІЧНИЙ ІНСТИТУТ(ВПІ)	023	Образотворче мистецтво, декоративне мис-	• Образотворче мистецтво

		тецтво, реставрація	
	061	Журналістика	• Реклама і зв'язки з громадськістю • Видавнича справа та редагування
	073	Менеджмент	• Медіаменеджмент і адміністрування в видавничо-поліграфічної галузі
	133	Галузеве машино-будування	• Комп'ютеризовані поліграфічні системи МС
	186	Видавництво та поліграфія	• Поліграфічні медіа технології МС • Технології електронних мультимедійних видань МС • Цифрові технології репродукування МС
ІНСТИТУТ АЕРОКОСМІЧНИХ ТЕХНОЛОГІЙ	134	Авіаційна та ракетно-космічна техніка	• Літаки та вертольоти • Ракетні та космічні комплекси
	173	Авіоніка	• Системи керування літальними апарата-ми та комплексами
ІНСТИТУТ ЕНЕРГОЗБЕРЕЖЕННЯ ТА ЕНЕРГОМЕНЕДЖМЕНТУ	141	Електроенергетика, електротехніка та електромеханіка ДП	• Системи електропостачання МС • Енергетичний менеджмент та енергоефективність МС • Інжиніринг автоматизова-них електротехнічних комплексів МС

			• Електромеханічні і мехатронні системи енергоємних виробництв МС
	144	Теплоенергетика ДП	• Енергетичний менеджмент і інжиніринг МС
	101	Екологія ДП	• Інженерна екологія та ресурсозбереження МС
	184	Гірнича справа ДП	• Розробка родовищ і видобуток корисних копалин МС • Геотехнічне і міське підземне будівництво МС
ІНСТИТУТ СПЕЦІАЛЬНОГО ЗВ'ЯЗКУ ТА ЗАХИСТУ ІНФОРМАЦІЇ	125	Кібербезпека	
	172	Телекомунікації та радіотехніка	
	122	Комп'ютерні науки	
МЕХАНІКО-МАШИНОБУДІВНИЙ ІНСТИТУТ	134	Авіаційна та ракетно-космічна техніка	• Літаки і вертольоти • Ракетна техніка
	173	Авіоніка	• Системи управління літальними апаратами і комплексами
	131	Прикладна механіки	• Динаміка і міцність машин • Інформаційні системи і технології авіабудуванні • Інженерія логістичних систем

			• Технології машинобудування • Технології виробництва літальних апаратів • Системи комп’ютерних технологій пластичного формоутворення в машинобудуванні • Технології композиційних і наноструктурних конструкцій • Технологія озброєння та засобів безпеки • Лазерна техніка і комп’ютеризованих процеси фізико-технічної обробки матеріалів • Лазерні системи в біології і медицині • Мехатронні системи в машинобудуванні • Гідравлічні та пневматичні машини і системи приводів • Технології комп'ютерного конструювання верстатів, роботів та машин • Автоматизовані та роботизовані механічні системи

			• Інструментальні системи і технології формоутворення деталей • Інженерний дизайн
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ	105	Прикладна фізика і наноматеріали ДП	• Високі фізичні технології • Фізика живих систем • Фізика нових джерел енергії
	113	Прикладна математика ІТ	• Математичні методи криптографічного захисту інформації • Математичні методи комп'ютерного моделювання
	125	Кібербезпека ІТ	• Системи і технології кібербезпеки • Математичні методи кібербезпеки • Системи технічного захисту інформації
ТЕПЛОЕНЕРГЕТИЧНИЙ ФАКУЛЬТЕТ	121	Інженерія програмного забезпечення	• Програмне забезпечення розподіле-них систем • Програмне забезпечення web-технологій і мобільних пристроїв

	122	Комп’ютерні науки	• Геометричне моделювання в інформаційних системах • Інформаційні технології моніторингу навколишнього середовища
	142	Енергетичне машинобудування	• Тепло і парогенеруючі установки
	143	Атомна енергетика	• Атомні електричні станції
	144	Теплоенергетика	• Промислова і муніципальна теплоенергетика та енергозбереження • Теплофізика • Теплові електричні станції та установки (МС)
	151	Автоматизація та комп’ютерно-інтегровані технології	• Автоматизоване управління технологічними процесами • Комп'ютерно-інтегровані технологічні процеси і виробництва

ЗВАРЮВАЛЬНИЙ ФАКУЛЬТЕТ	131	Прикладна механіка	• Технології спеціальні види робіт у зварюванні МС • Автоматизовані технологічні системи в зварюванні МС • Споріднені технологія зварювання і ресурсозбереження МС
ІНЖЕНЕРНО-ФІЗИЧНИЙ ФАКУЛЬТЕТ	132	Матеріалознавство	• Матеріалознавство порошкових композиційних матеріалів і покриттів • Нанотехнології і комп'ютерний дизайн матеріалів • Металознавство і процеси термічної обробки фізичне матеріалознавство
	136	Металургія	• Ливарне виробництво і комп'ютеризація процесів лиття • Художнє і ювелірне литво • Спеціальна металургія • Порошкова металургія
ІНЖЕНЕРНО-ХІМІЧНИЙ ФАКУЛЬТЕТ	101	Екологія	• Екологічна безпека
	131	Прикладна механіка	• Інжиніринг, комп'ютерне моделювання та проекту-вання обладнання упаковки МС

	133	Галузеве машинобудування	• Інжиніринг, комп'ютерне моделювання та проекту-вання обладнання виробництв полімерних і будівель-них матеріалів і виробів МС • Інжиніринг, комп'ютерне моделювання та проекту-вання обладнання целюлозно-паперового виробництва МС • Інжиніринг, комп'ютерне моделювання та проектування обладнання хімічних і нафтопереробних виробництв
	151	Автоматизація та комп’ютерно-інтегровані технології	• Автоматизація хіміко-технологічних процесів і виробництв МС • Комп'ютерно-інтегровані технології хімічних і нафтопереробних виробництв
	161	Хімічні технології та інженерія	• Хімічні технології переробки деревини та рослинної сировини МС
ПРИЛАДОБУДІВНИЙ ФАКУЛЬТЕТ	151	Автоматизація та комп'ютерно-інтегровані технології	• комп'ютерно-інтегровані технології виробництва при-ладів (МС)

			• комп'ютерно-інтегровані технології приладів точної механіки (МС) • комп'ютерно-інтегровані технології та системи навігації та управління (МС) • комп'ютерно-інтегровані технології та системи неруйнівного контролю і діагностики комп'ютерно-інтегровані оптико-електронні системи та технології (МС)
	152	Метрологія та інформаційно-вимірювальна техніка	• Медичні прилади та системи (МС) • Інформаційні технології та вимірювальні системи точної механіки • Інформаційно-вимірювальні технології екологічного моніторингу (МС)) • Фотоніка та оптоінформатіка (МС)
РАДІОТЕХНІЧНИЙ ФАКУЛЬТЕТ	172	Телекомунікації та радіотехніка	• Інтелектуальні технології мікросистемної радіоелектронної техніки • Радіозв'язок і обробка сигналів • Радіосистемна інженерія

			• Радіотехнічні інформаційні технології
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ	122	Комп'ютерні науки ІТ	• Інформаційні технології в біології та медицині
	227	Фізична терапія, Ерготерапія	• Фізична терапія
	163	Біомедицинська інженерія ДП	• Реабілітаційна інженерія • Клінічна інженерія • Біомедицинська інформатика • Регенеративне і біофармацевтичних інженерія
	152	Метрологія та інформаційно-вимірювальна техніка ДП	• Біомедицинські прилади та інформаційні системи
ФАКУЛЬТЕТ БІОТЕХНОЛОГІЇ І БІОТЕХНІКИ	133	Галузеве машинобудування ДП	• Обладнання фармацевтичних та біотехнологічних виробництв
	162	Біотехнології та біоінженерія ДП	• Промислова біотехнологія • Молекулярна біотехнологія • Екологічна біотехнологія та біоенергетика
ФАКУЛЬТЕТ ЕЛЕКТРОЕНЕРГОТЕХНІК І АВТОМАТИКИ	141	Електроенергетика, електротехніка та електромеханіка ДП	• Електричні станції МС • Електричні системи і мережі МС • Техніка і електрофізики високих напруг

			• Системи управління виробництвом і розподілом електроенергії МС • Нетрадиційні і поновлювані джерела енергії • Електричні машини і апарати МС • Електромеханічні системи автоматизації та електропривод МС
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ	153	Мікро- та наносистемної техніка ДП	• Мікро- та наноелектронні прилади і пристрої • Мікроелектронні інформаційні системи • Електронні біомедичні системи і технології • Інформаційні технології проектування в електроніці та наносистемах
	171	Електроніка ДП	• Електронні прилади та пристрої МС • Електронні компоненти й системи МС • Акустичні мультимедійні технології і системи • Акустичний моніторинг, біо- і Психоакустика • Електронні та інформаційні технології кінематографії та аудіовізуальних систем МС

	172	Телекомунікації та радіотехніка ДП	• Інформаційно-обчислювальні засоби електронних систем
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ	121	Інженерія програмного забезпечення	• Програмне забезпечення інформацій-них управляючих систем та технологій • Програмне забезпечення високопродуктивних комп'ютер-них систем і мереж • Програмне забезпечення інформацій-но-комунікаційних систем • Програмне забезпечення інтелектуальних і робототехнічних систем
	123	Комп'ютерна інженерія	• Комп'ютерні системи і мережі • Технології програмування для комп'ютерних систем і мереж
	126	Інформаційні системи і технології	• Інтегровані інформаційні системи • Інформаційне забезпечення робототех-нічних систем • Інформаційні управляючі системи та технології

ФАКУЛЬТЕТ ЛІНГВІСТИКИ	035	Філологія	• Германські мови і література • Романські мови та література
ФАКУЛЬТЕТ МЕНЕДЖМЕНТУ ТА МАРКЕТИНГУ	051	Економіка	• Економічна кібернетика • Міжнародна економіка • Економіка підприємства • Управління персоналом і економіка праці • Бізнес-аналітика
	075	Маркетинг	• Промисловий маркетинг
	073	Менеджмент	• Менеджмент і бізнес-адміністрування • Менеджмент інвестицій та інновацій • Менеджмент міжнародного бізнесу • Логістика
ФАКУЛЬТЕТ СОЦІОЛОГІЇ І ПРАВА	054	Соціологія	• Врегулювання конфліктів та медіація
	081	Право	• Господарське та адміністративне право і процес • Інформаційне право та право інтелектуальної власності
	231	Соціальна робота	• Міжнародні соціальні проекти і волонтерська діяльність
	281	Публічне управління і адміністрування	• Адміністративний менеджмент • електронне урядування

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ	113	Прикладна математика ІТ	• Наука про дані і математичне моделювання
	123	Комп'ютерна інженерія ІТ	• Комп'ютерні системи та компоненти • Системне програмування • Спеціалізовані комп'ютерні системи
	121	Інженерія програмного забезпечення ІТ	• Програмне забезпечення комп'ютерних та інформаційно-пошукових систем
ФІЗИКО-МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ	111	Математика ДП	• Страхова і фінансова математика • Математичні і комп'ютерні методи в моделюванні динамічних систем
	104	Фізика та астрономія ДП	• Комп'ютерне моделювання фізичних процесів
ХІМІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ	161	Хімічні технології та інженерія	• Електрохімічні технології неорганічних і органічних матеріалів • Хімічні технології косметичних засобів і харчових добавок • Хімічні технології неорганічних і органічних сполучних і композиційних матеріалів Хімічні технології неор-ганічних керамічних матеріалів

			• Хімічні технології неорганічних речовин та водоочищення • Хімічні технології органічних речовин

3 ЗАСОБИ РОЗРОБКИ

3.1 Вибір СУБД

Вибір СУБД, як засоби для створення функціювання бази даних автоматизованої системи, здійснюється на основі аналізу ряд характеристик сучасних СУБД, представлених в таблиці 1.

Таблиця – Порівняльна характеристика СУБД

Функція	SQL Server 2008	Oracle	MySQL	Access	Visual FoxPro
Операційна система	Windows 2000 або NT і вище	OS/2, UNIX, Windows, MS-DOS	Windows NT і вище	Windows 95 і вище	Windows 95 або NT
Підтримка SQL	+	+	+	+	+
Підтримка функцій ODBC	+	+	+	+	-
Підтримка Visual Basic	-	-	-	+	
Підтримка COM	+	+		+	+
Представлення (Views)	+	-	+	-	-
Індексоване представлення	+	-	-	-	-
Використання OLE DB	+	+	-	+	+
Інтеграція з пакетом Office 2000	+	-	-	+	-
Засоби створення форм і звітів	-	-	-	+	+
Засоби аналізу даних OLAP	+	+	-	-	-
Копіювання БД	+	+	-	+	+

Розвиток підтримки XML	+	+	-	-	-
Безпека	+	+	-	-	-
Адміністрування бази даних	+	+	-	+	-
Доступ до даних через інтернет	+	+	+	+	+
Імпорт і експорт	+	+	-	+	+
Підтримка JDBC - драйвера	+	+	+	-	-
Збережені процедури	+	+	-	-	-
Трігери	+	+	-	-	-

Згідно представлених даних призводить до необхідності використання такого додатку, інтерфейс і збереження даних якого би в базі формату Microsoft SQL.

3.2 Вибір і обґрунтування мови та середовища програмування

В даний час існує велика кількість мов програмування, з якими можна працювати, у кожної з якої є переваги і недоліки, які виявляються в процеси роботи з ними.

Правильний вибір мови програмування допоможе створите просте рішення з можливостю налагодження, документування і виправлення помилок.

Для створення зручної і багатофункціональної інформаційної системи найкраще використовувати мову програмування, яку можна зв’язувати з іншими мовами і при цьому зв’язувати програму з базами даних.

Borland Delphi — інтегрована система розробки програмного забезпечення для Microsoft Windows, Mac OS, iOS і Android на мові Delphi. Середовище назначено для швидкої (RAD) розробки прикладного програмного забезпечення для операційних систем Windows, Mac OS X, а також IOS і Android. Завдяки унікальної сукупності легкості мови і генерації машинного коду, дозволяє безпосередньо, і, при бажанні,

досить низькорівнево взаємодіяти з операційною системою, а також з бібліотеками, написаними на C/C++. Створення програми незалежної від стороннього програмного забезпечення, як наприклад Microsoft.NET Framework, або Java Virtual Machine.

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читаності коду. Синтаксис ядра Python мінімалістичний. В той же час стандартна бібліотека включає великий об’єм корисних функцій.

C# — об’єктно-орієнтована мова програмування. Розроблена в 1998—2001 роках групою інженерів під керівництвом Андерса Хейлсберга в компанії Microsoft як мова розробки додатків для платформи Microsoft .NET Framework.

C# відноситься до C-подібних мов за синтаксисом. Його синтаксис найбільш подібний до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML, LINQ, виключення, коментарі в формате XML.

C# — відносно нова об'єктно-орієнтирована мова програмування, розроблена компанією Microsoft. На ній можна писати додатки для Web, додатки під Windows, консольні програми, які запускаються з командною строкою, працювати з базами данними.[10].

C# - це лише одна з мов платформи .NET Framework (крім Visual Basic.NET, Visual C++.NET, J#, а з появленням Delphi 8/2005 ще і Delphi). В цілому C# ввібрав в себе багато чого з того кращого, що є в багатьох мов програмування. Це проста, надійна, повністю об'єктно орієнтована, використана платформою .NET Framework.

Платформа .NET - це нова технологія розробки програмного забезпечення. В її основі лежить ідея універсального програмного коду, який зможе працювати на любому пристрою, незалежно від встановленої на цій системі операційної системи (ОС повинна підтримувати технологію .NET). Універсальність програмного коду забезпечується за рахунок попередньої, на етапі розробки, компіляції вихідної

програми в проміжний код на мові CIL (Common Intermediate Language), який під час завантаження транслюється в виконавчу програму. Платформа .NET є незалежною від використаних мов програмування. Можна використовувати декілька .NET-сумісних мов навіть в рамках одного проекту.

Дана мова є найбільш простою і привабливою для початківців програмістів, ніж C++. На C# можна написати що завгодно: веб-сервіси, мобільні ПО, серверні додатки і так далі. ЇЇ основна середа програмування, в якій використовується, Visual Studio – дуже зручна і зрозуміла середа розробки додатків.

В комплект Visual Studio входять наступні основні компоненти:

1. Visual C# - на мові C# (Microsoft);

2. Visual F# - на F# (Microsoft Developer Division).

3. Visual Basic.NET - для розробки додатків на Visual Basic;

4. Visual C++ - на традиційній мові C++;

Функціональна структура середи включає в себе:

– редактор вихідного кода;

– редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;

– веб-редактор;

– відкладчик коду;

– дизайнер схем баз даних;

– дизайнер класів.

Visual Studio також дозволяю створювати і підключити стороні доповнення (плагіни) для розширення функціональності на кожному рівні, включає додавання підтримки систем контроля версій вихідного коду, додавання нових наборів інструментів (для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесів розробки програмного забезпечення).

Висновок: Так як випускна кваліфікаційна робота націлена на розробку продукту, що має гнучкий графічний інтерфейс, основним завданням якого є робота з базою даних, вибирається мова C #. Проект буде реалізований в середовищі програмування Visual Studio 2019.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Аналіз поточного стану проблеми

Поточна організація реєстрації абітурієнтів на подачу документів наведена на рис.1

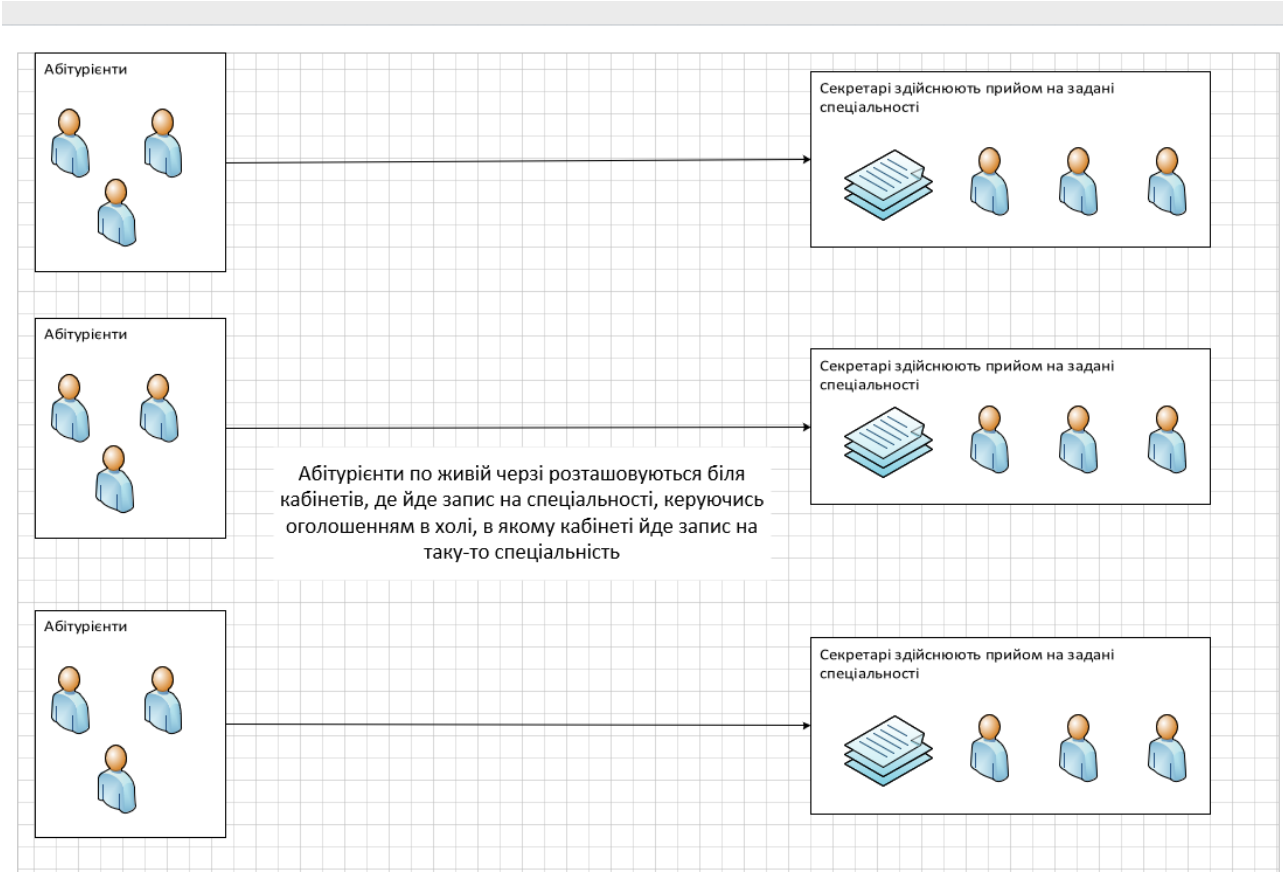

Рис.1 – Поточна організація реєстрації абітурієнтів

Дана організація має багато недоліків:

- неможливість відновлення в черзі в разі відсутності;
- обов’язковість присутності в черзі;
- архаїчність – в епоху інформаційних технологій такий спосіб запису абітурієнтів у одно з кращих технічно-інформаційних університетів країни просто не допустимо.

4.2 Пропоноване рішення проблеми

Пропонована система реєстрації абітурієнтів на основі системи моніторингу Електронної Черги вступників КПІ приведена на малюнку 2

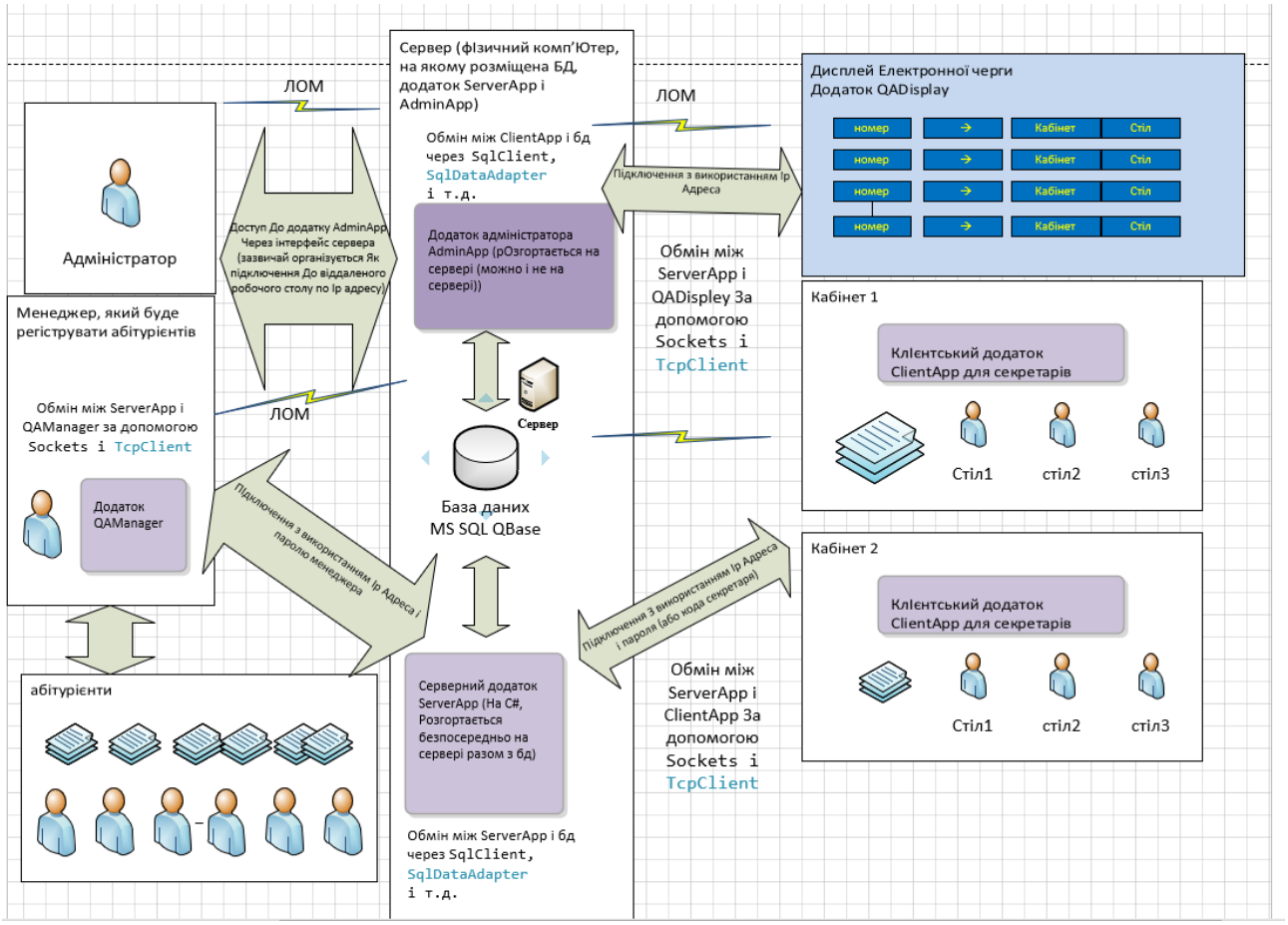

Рис.2 – Пропонована система реєстрації абітурієнтів

Виходячи із запропонованої схеми необхідно розробити такі елементи та додатки:

- базу даних в форматі MS SQL SERVER QBase.mdf.
- ServerApp – сервер;
- ClientApp – додаток секретаря;

- QAManager – додаток менеджера, реєструючого абітурієнтів; людина, яка приймає абітурієнтів і назначає їм номер (видає талон);
- AdminApp – додаток для адміністрування БД.
- QADisplay – додаток для виводу поточної електронної черги на екран.

4.3 Проектування реляційної бази даних

4.3.1 Розробка ER-моделі

Перш ніж розробляти ER-модель предметної області, необхідно виділити основні сутності, характерні для даної предметної області, визначається інформація, яка повинна зберігатися в БД.

Виділяються базові сутності цієї предметної області:

Сутності: Кабінет, Спеціальність, Факультет, Поточна черга, Налаштування, Спеціалізація, Користувач, Робочий стіл.

Далі будується ER–діаграма, що містить атрибути сутності. Виділяється атрибути сутності цієї предметної області, складається словесний опис:

- Сутність **Кабінет**: *Код Кабінету*, Назва кабінету;
- Сутність **Спеціальності**: *Код спеціальності*, Назва спеціальності, Код спеціалізації;
- Сутність **Факультет**: *Код факультету*, Назва факультету;
- Сутність **Поточна черга**: id черги, номер черги, id столу, id спеціальності, час реєстрації, час виклику, час обслуговування, час завершення обслуговування, час відкладення, час чекання, id користувача - секретаря, id користувача - менеджера, номер статусу, id спеціалізації, ФИО абітурієнта, дата народження абітурієнта, ФДП, МАН, ВУО, ОЛИМП;
- Сутність **Налаштування:** id налаштування, максимальна кількість черг;

- Сутність **Спеціалізації**: id спеціалізації, назва спеціалізації, Код спеціалізації, id факультету;
- Сутність **Користувача**: id користувача, Логін, Пароль, ФІО, e-mail, ознака адміністратора, ознака працюючого, телефон;
- Сутність **Робочий стіл**: id номінації, Назва столу, id кабінету.

Детальні характеристики даталогічної моделі відображено в таблиці 3. Ключові поля виділені жирним курсивом. Імена полям бази даних даються на латинице.

Таблиця 1 – Даталогічна модель

Об’єкт	Характеристика	Тип інформації	Примітка
Faculties	***idFaculty***	*[int] IDENTITY(1,1)*	*Ключове поле (NOT NULL)*
	nameFaculty	[nvarchar](250)	NOT NULL
Cabinet	***idCabinet***	*[int] IDENTITY(1,1)*	*Ключове поле(NOT NULL)*
	nameCabinet	[nvarchar](10)	
settingsQueue	***idSetting***	*[int] IDENTITY(1,1)*	*Ключове поле(NOT NULL)*
	countMaxQueue	int	NOT NULL
Users	***IdUser***	*[int] IDENTITY(1,1)*	*Ключове поле(NOT NULL)*
	Login	[nvarchar](20)	NOT NULL
	Password	[nvarchar](255)	NOT NULL
	FullName	[nvarchar](255)	NOT NULL
	MailAdres	[nvarchar](255)	NULL
	Admin	[bit]	NOT NULL
	Works	[bit]	NOT NULL
	Telefon	[nvarchar](50)	NULL
WTable	***idTable***	*[int] IDENTITY(1,1)*	*Ключове поле (NOT NULL)*
	nameTable	[nvarchar](10)	NOT NULL
	idCabinet	[int]	NOT NULL

Specialty	IdSpecialty	[int] IDENTITY(1,1)	*Ключове поле(NOT NULL)*
	NameSpecialty	[nvarchar](255)	NOT NULL
	KodSpecialty	[nvarchar](10)	NULL
	idFaculty	int	NULL
SpecialtyOnTable	***idSpecialtyOnTable***	[int] IDENTITY(1,1)	*Ключове поле(NOT NULL)*
	idSpecialty	int	NOT NULL
	idTable	int	NOT NULL
	Priority	int	NOT NULL
DetailSpecialty	***idDetailSpecialty***	[int] IDENTITY(1,1)	*Ключове поле(NOT NULL)*
	nameDetailSpecialty	[nvarchar](250)	NOT NULL
	idSpecialty	Int	NOT NULL
QueueCurrent	***idQueueCurrent***	[int] IDENTITY(1,1)	*Ключове поле(NOT NULL)*
	Number	int	NOT NULL
	idTable	int	NULL
	idSpecialty	int	NOT NULL
	timeRegistration	smalldatetime	NULL
	timeCall	smalldatetime	NULL
	timeServiceS	smalldatetime	NULL
	timeEndService	smalldatetime	NULL
	timePutOff	smalldatetime	NULL
	timeWaitingS	smalldatetime	NULL
	idUser	int	NULL
	idManager	int	NULL
	intStatus	int	NULL
	idSubSpeciality	int	NULL
	nameClient	[nvarchar](100)	NULL
	dateBirthClient	date	NULL
	FDP	[nvarchar](3)	NULL
	MAN	[nvarchar](3)	NULL
	VUO	[nvarchar](3)	NULL
	OLIMP	[nvarchar](3)	NULL

4.3.2 Розробка логічної і фізичної схеми бази даних.

Для розробки логічної і фізичної схеми бази даних, генерації скриптів для створення БД скористуємося програмою AIIFusion ERwin Data Modeler by CA. Дана програма дозволяє згенерувати sql-скрипт для створення об'єктів бази даних в форматі MS SQL. Логічна модель бази даних представлена на рис.3

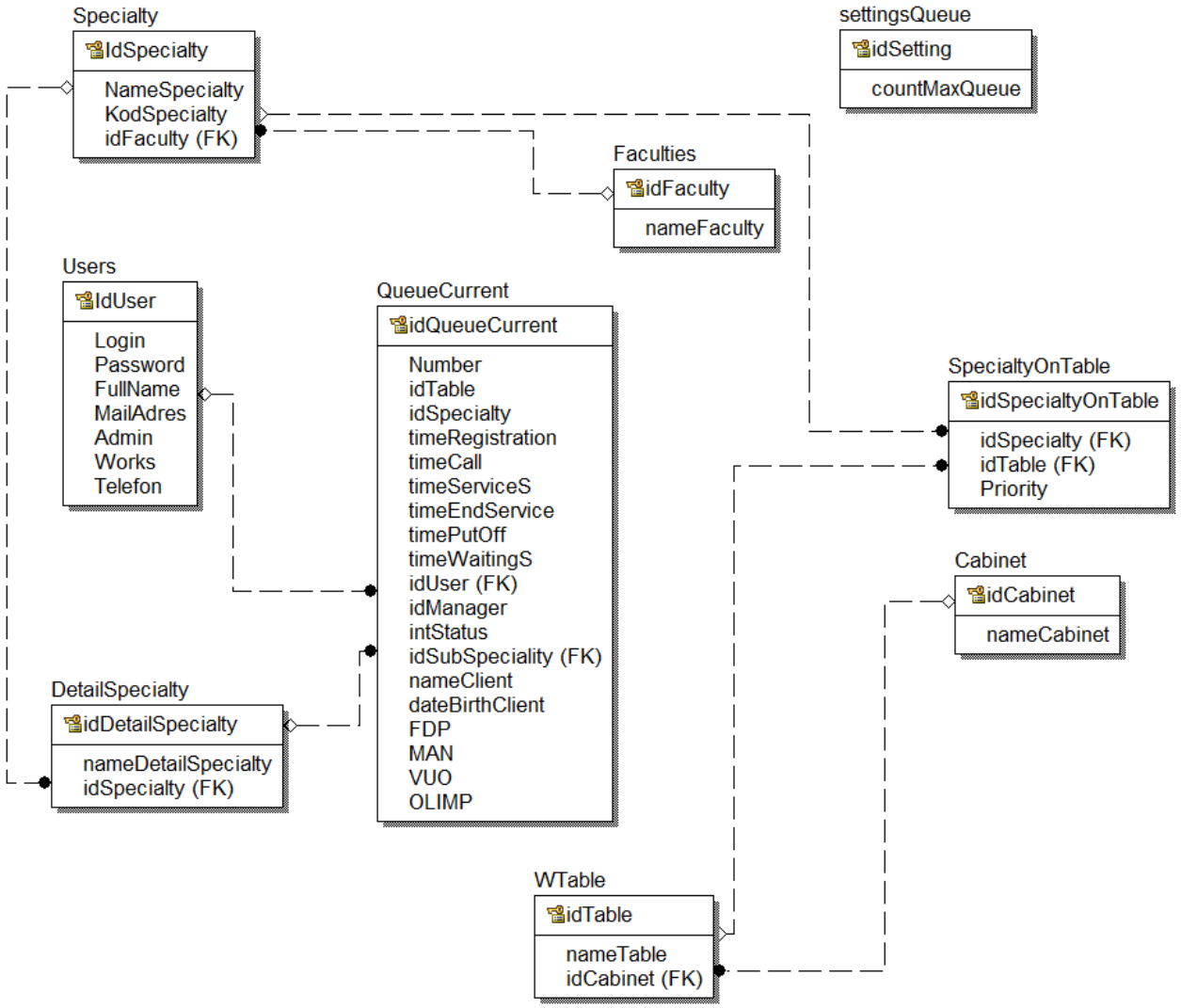

Рис.3 – Логічна модель бази даних

Фізична модель бази даних представлена на рис.4

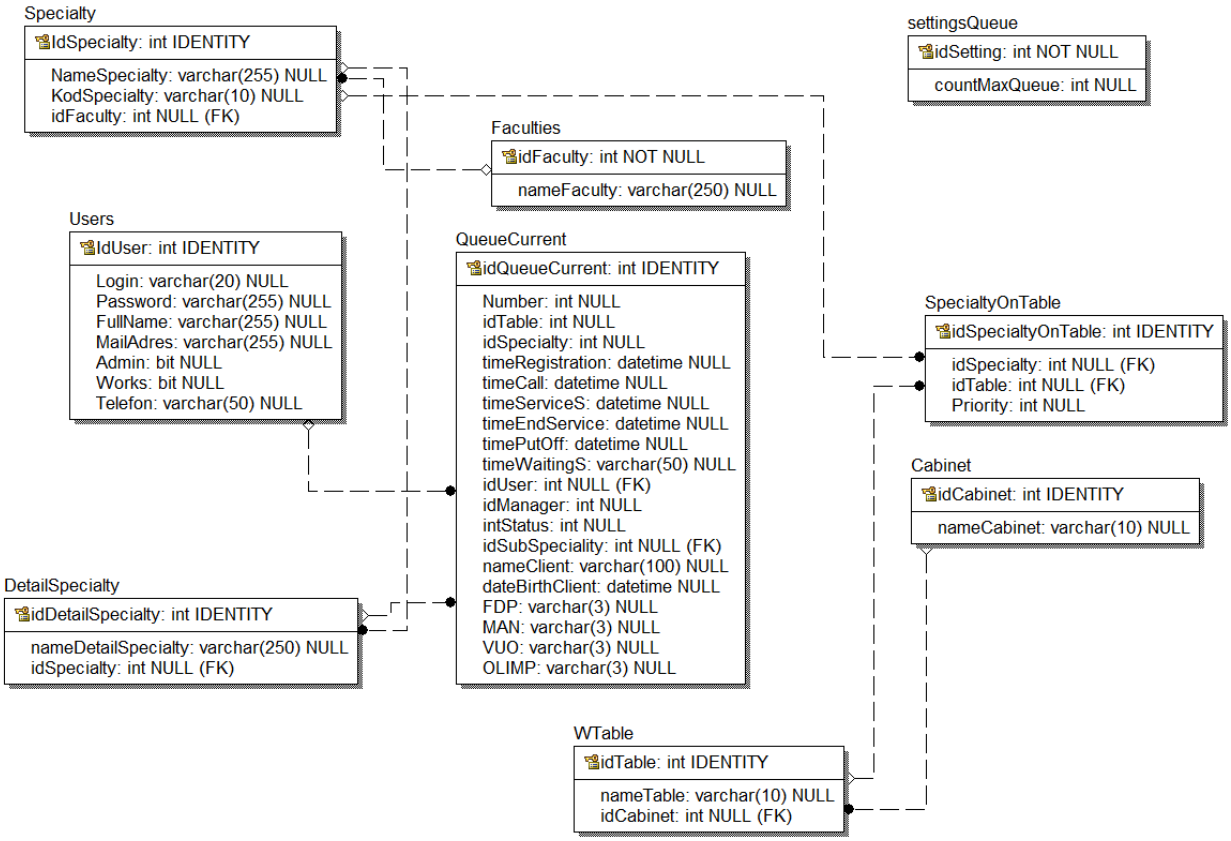

Рис.4 – Фізична модель бази даних

4.4 Створення реляційної бази даних MS SQL

Після успішного проектування бази даних створим физичну базу даних Qbase.mdf.

В програмі AIIFusion ERwin Data Modeler by CA згенеруємо sql-скрипт для створення бази даних. Послідовність генерації представлена на малюнках

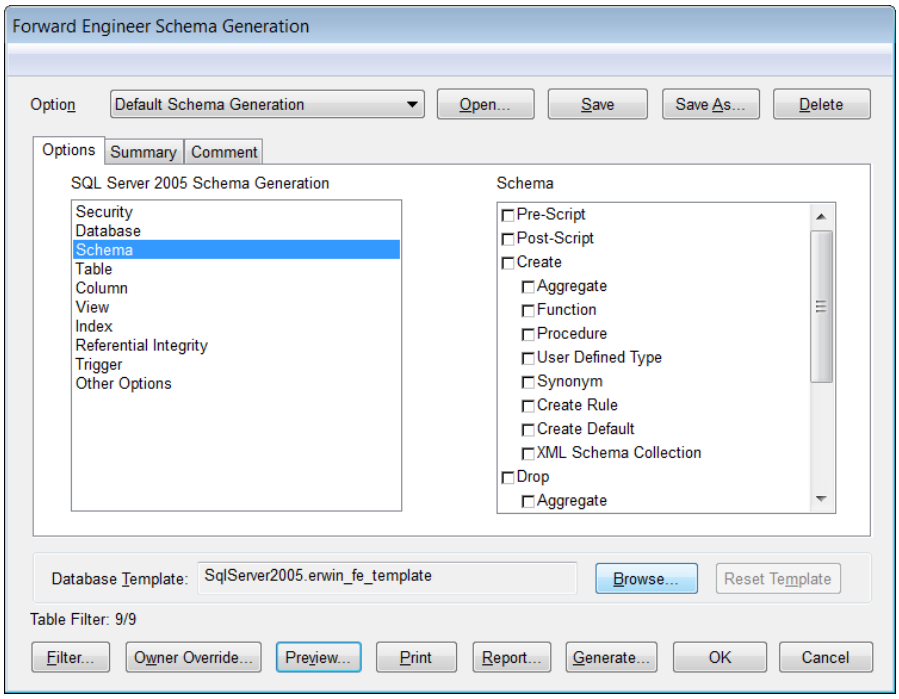

Рис.5 – Вибір даних для генерації скрипту

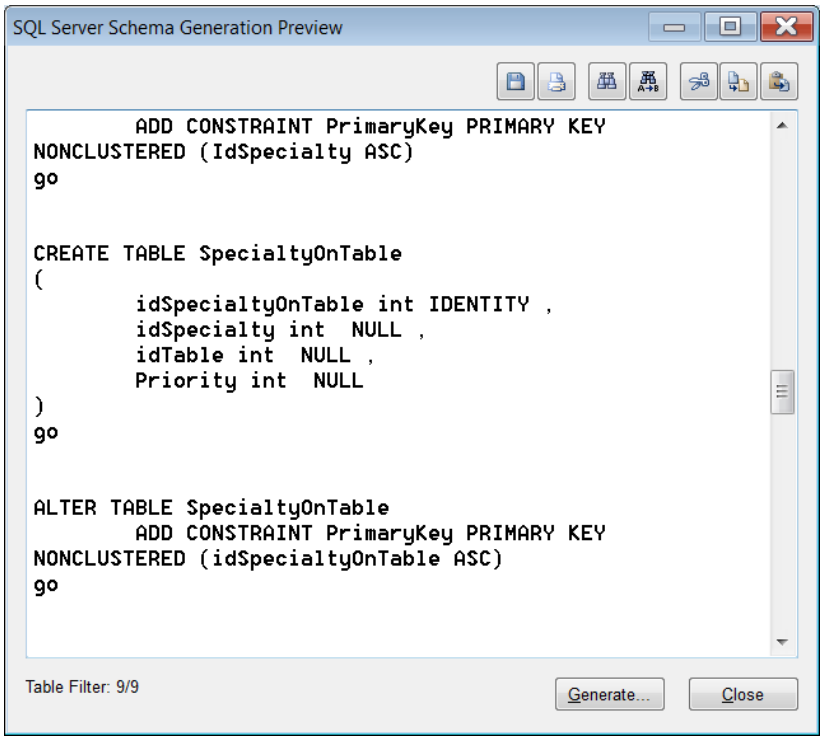

Рис.6 – згенерований скрипт.

Скрипт для створення БД представлено нижче

```
CREATE TABLE Cabinet
(
      idCabinet int IDENTITY ,
      nameCabinet varchar(10)  NULL
)
go
ALTER TABLE Cabinet
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (idCabinet ASC)
go
CREATE TABLE DetailSpecialty
(
      idDetailSpecialty int IDENTITY ,
      nameDetailSpecialty varchar(250)  NULL ,
      idSpecialty int  NULL
)
go
ALTER TABLE DetailSpecialty
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (idDetailSpecialty
ASC)
go
CREATE TABLE Faculties
(
      idFaculty int  NOT NULL ,
      nameFaculty varchar(250)  NULL
)
go
ALTER TABLE Faculties
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (idFaculty ASC)
go
CREATE TABLE QueueCurrent
(
      idQueueCurrent int IDENTITY ,
      Number int  NULL ,
      idTable int  NULL ,
      idSpecialty int  NULL ,
      timeRegistration datetime  NULL ,
      timeCall datetime  NULL ,
      timeServiceS datetime  NULL ,
      timeEndService datetime  NULL ,
      timePutOff datetime  NULL ,
      timeWaitingS varchar(50)  NULL ,
      idUser int  NULL ,
      idManager int  NULL ,
      intStatus int  NULL ,
      idSubSpeciality int  NULL ,
      nameClient varchar(100)  NULL ,
      dateBirthClient datetime  NULL ,
      FDP varchar(3)  NULL ,
      MAN varchar(3)  NULL ,
      VUO varchar(3)  NULL ,
      OLIMP varchar(3)  NULL
)
go
```

```
ALTER TABLE QueueCurrent
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (idQueueCurrent
ASC)
go
CREATE TABLE settingsQueue
(
      countMaxQueue int  NULL ,
      idSetting int  NULL ,
      IdSettings int IDENTITY (1,1)
)
go
ALTER TABLE settingsQueue
      ADD CONSTRAINT XPKsettingsQueue PRIMARY KEY  NONCLUSTERED (IdSettings
ASC)
go
CREATE TABLE Specialty
(
      IdSpecialty int IDENTITY ,
      NameSpecialty varchar(255)  NULL ,
      KodSpecialty varchar(10)  NULL ,
      idFaculty int  NULL
)
go
ALTER TABLE Specialty
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (IdSpecialty ASC)
go
CREATE TABLE SpecialtyOnTable
(
      idSpecialtyOnTable int IDENTITY ,
      idSpecialty int  NULL ,
      idTable int  NULL ,
      Priority int  NULL
)
go
ALTER TABLE SpecialtyOnTable
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (idSpecialtyOnTable
ASC)
go
CREATE TABLE Users
(
      IdUser int IDENTITY ,
      Login varchar(20)  NULL ,
      Password varchar(255)  NULL ,
      FullName varchar(255)  NULL ,
      MailAdres varchar(255)  NULL ,
      Admin bit  NULL ,
      Works bit  NULL ,
      Telefon varchar(50)  NULL
)
go
ALTER TABLE Users
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (IdUser ASC)
go
```

```
CREATE TABLE WTable
(
      idTable int IDENTITY ,
      nameTable varchar(10)  NULL ,
      idCabinet int  NULL
)
go
ALTER TABLE WTable
      ADD CONSTRAINT PrimaryKey PRIMARY KEY  NONCLUSTERED (idTable ASC)
go
ALTER TABLE DetailSpecialty
      ADD CONSTRAINT  SpecialtyDetailSpecialty FOREIGN KEY (idSpecialty)
REFERENCES Specialty(IdSpecialty)
            ON DELETE NO ACTION
            ON UPDATE CASCADE
go
ALTER TABLE QueueCurrent
      ADD CONSTRAINT  DetailSpecialtyQueueCurrent FOREIGN KEY
(idSubSpeciality) REFERENCES DetailSpecialty(idDetailSpecialty)
            ON DELETE NO ACTION
            ON UPDATE CASCADE
go
ALTER TABLE QueueCurrent
      ADD CONSTRAINT  UsersQueueCurrent FOREIGN KEY (idUser) REFERENCES
Users(IdUser)
            ON DELETE NO ACTION
            ON UPDATE CASCADE
go
ALTER TABLE Specialty
      ADD CONSTRAINT  FacultiesSpecialty FOREIGN KEY (idFaculty) REFERENCES
Faculties(idFaculty)
            ON DELETE NO ACTION
            ON UPDATE CASCADE
go
ALTER TABLE SpecialtyOnTable
      ADD CONSTRAINT  SpecialtySpecialtyOnTable FOREIGN KEY (idSpecialty)
REFERENCES Specialty(IdSpecialty)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE SpecialtyOnTable
      ADD CONSTRAINT  WTableSpecialtyOnTable FOREIGN KEY (idTable) REFERENCES
WTable(idTable)
            ON DELETE NO ACTION
            ON UPDATE CASCADE
go
ALTER TABLE WTable
      ADD CONSTRAINT  CabinetWTable FOREIGN KEY (idCabinet) REFERENCES
Cabinet(idCabinet)
            ON DELETE NO ACTION
            ON UPDATE CASCADE
go
```

В програмі SQL Server Management Studio створюємо базу даних Qbase. Виконуємо скрипт, згенерований в програмі AIIFusion ERwin Data Modeler by CA. Створено файл Qbase.mdf. Діаграма бази даних Qbase.mdf представлена на рис. 7.

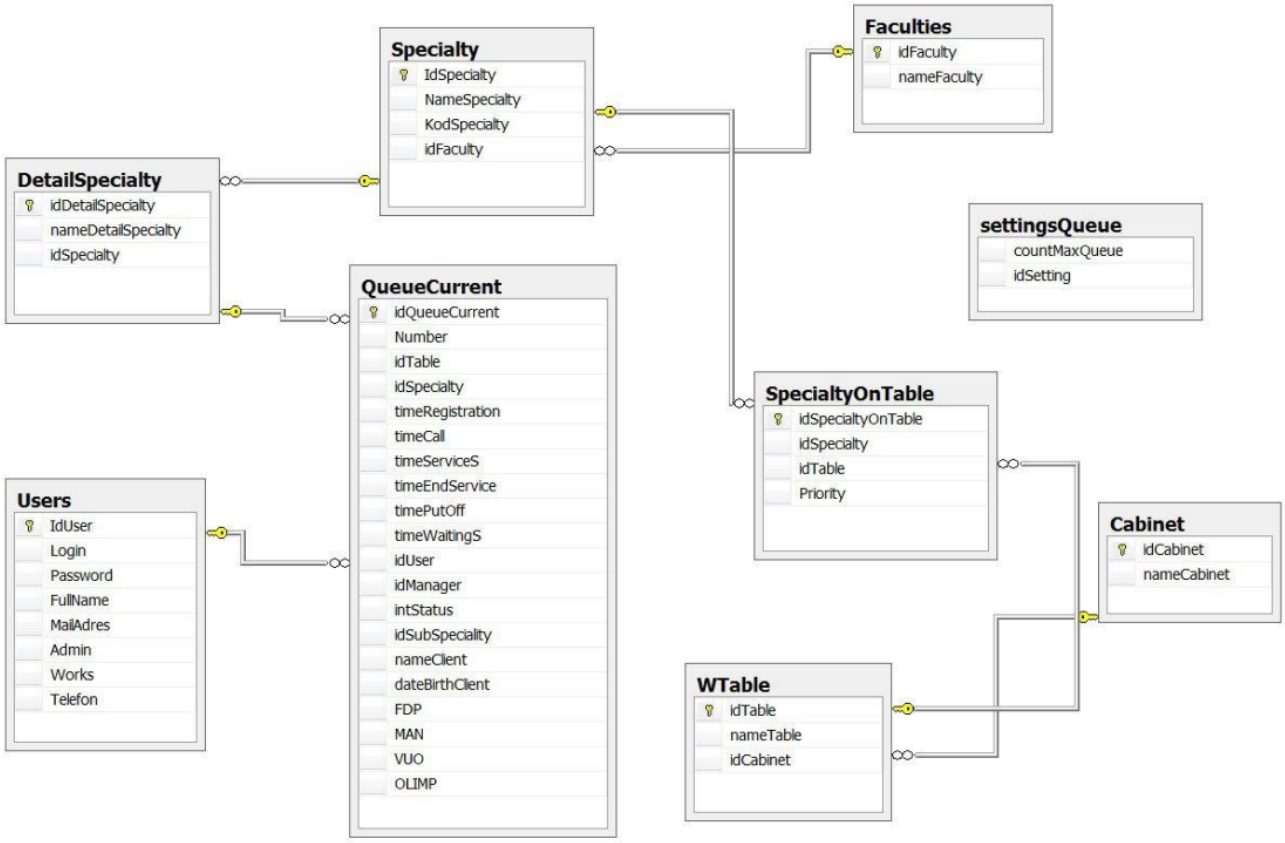

Рис.7 – Діаграма бази даних Qbase.mdf

4.5 Призначення системи моніторингу електронної черги абітурієнтів КПІ

Система моніторингу електронної черги абітурієнтів КПІ призначена для вдосконалення обслуговування абітурієнтів в КПІ під час приймальної компанії. Система дозволяє абітурієнту не стояти в декількох чергах випадку вступу на декілька спеціальностей. Забезпечена можливість стояти одночасно в декількох чергах.

Основні функції системи:

- Черга повинна бути одна для одного абітурієнта;
- Абітурієнт отримує інформацію з дисплею електронної черги про те, в якому кабінеті і на якому робочому місці йому необхідно прибути;
- Секретарі, які опрацьовують заявки, забезпечені зручною системою вибору абітурієнта для обслуговування;
- Дисплей електронної черги забезпечує зручний спосіб відображення черги абітурієнтів;
- Реалізована можливість закріплення пріоритетних і не пріоритетних спеціальностей;

4.6 Алгоритм роботи Системи моніторингу електронної черги вступників КПІ

- Абітурієнт підходить до менеджеру черги, у якого на робочому місці розгорнуто додаток Qmanager;
- Називає менеджеру черги перелік спеціальностей (максимальне число черг обмежено налаштуванням системи), менеджер обирає в додатку Qmanager необхідну спеціальність і видає абітурієнту напечатаний талон з наступними реквізитами:

1. Номер талона

2. Прізвище абітурієнта
3. Факультет
4. Направлення
5. Спеціальність
6. Дата і час реєстрації;

- Абітурієнт чекає, коли номер його талона висвітиться на центральному інформаційному таблі.
- Поруч з номером талона спочатку може бути не вказаний номер кабінета і робоче місце, в якому випадку підсвічування буде мати темно синій колір;
- Секретар бачe в вікні програми обслуговування абітурієнтів ClientApp, що абітурієнт чекає, і натискає кнопку виклику;
- Підсвічування номера талона абітурієнта і номера робочого місця і кабінету, в якому запрошується абітурієнт, на центральному інформаційному таблі прийме світло-синій колір;
- Абітурієнт підходить до «свого» робочого місця і подає талон;
- Інформація з великих дисплеїв стирається тільки після того, як вже заповнені всі рядки.

4.6 Компоненти системи моніторингу управління електронної чергою КПІ

1. Великий монітор, змонтований на стінку або підвішений до стелі (кількість дисплеїв і рядків залежить від кількості ПК на яких можна запустити додаток **Qdisplay**);
2. Комп’ютер і принтер, який печатає талони на робочому місці менеджера, на якому запущено додаток **Qmanager**;
3. Комп’ютер адміністратора, на якому запущено додаток **AdminApp**;

4. Локальна мережа для об’єднання всіх ПК системи й зв’язку з сервером електронної черги – додаток **ServerApp**;

5. Комунікаційні дроти і дроти живлення;

6. Сервер MS SQL на якому розміщена база даних **QBase.mdf**;

7. Комп’ютерна програма для кожного робочого місця для секретаря (програмний пульт обслуговування абітурієнтів **ClientApp**).

Розроблені додатки і схема взаємозв’язку додатків між собою і сервером і з базою даних відображена на рис. 8.

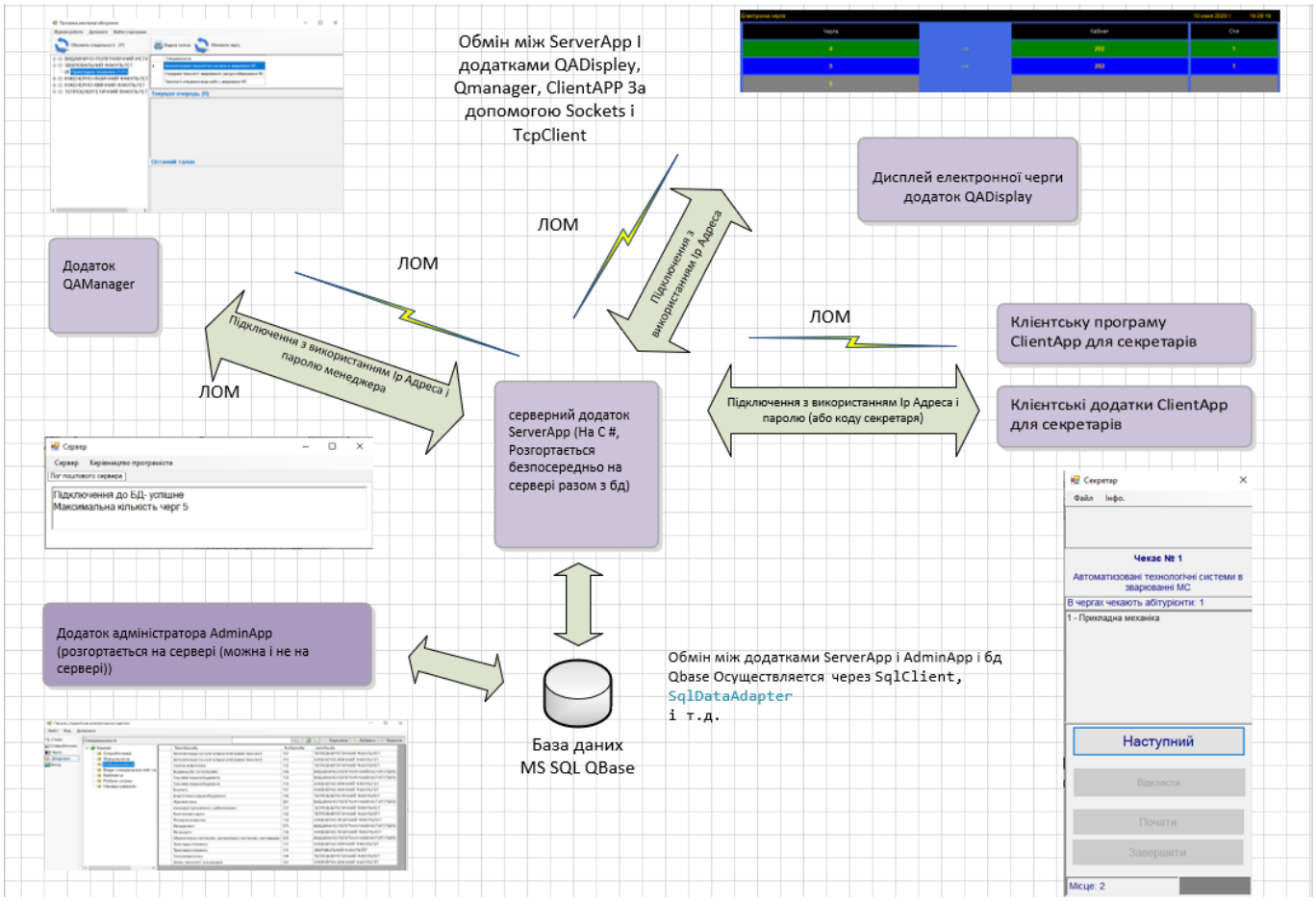

Рис.8 – Схема взаємозв’язку додатків між собою і сервером і з базою даних

В ході реалізації проекту були створені наступні проекти.

– база даних в формате MS SQL SERVER QBase.mdf (таблиці WTable, Specialty, SpecialtyOnTable, DetailSpecialty, QueueCurrent, Faculties, Cabinet, settingsQueue, Users);

– проект ServerApp.sln (додаток ServerApp.exe) – сервер;

– проект ClientApp.sln (додаток ClientApp.exe) – додаток секретаря;

– проект QAManager.sln (додаток QAManager.exe) – додаток менеджера, реєструючого студентів, людина яка приймає студентів і призначає їй номер (видає талон);

– проект AdminApp.sln (додаток AdminApp.exe) – додаток для адміністрування БД.

– проект QADisplay.sln (додаток QADisplay.exe) – додаток для виведення поточної електронної черги на екран.

4.7 Додаток ServerApp.exe

Додаток ServerApp.exe призначений для обробки запитів від клієнтських додатків (QADisplay.exe, ClientApp.exe, QAManager.exe) шляхом отримання необхідних відомостей з бази даних Qbase.mdf і відправлення відповіді на запит клієнтського додатку. Додаток ServerApp.exe повинен запускатися першим. Вигляд додатку у випадку успішного і неуспішного підключення до БД приведено на рис. 4.7.1 - 4.7.2.

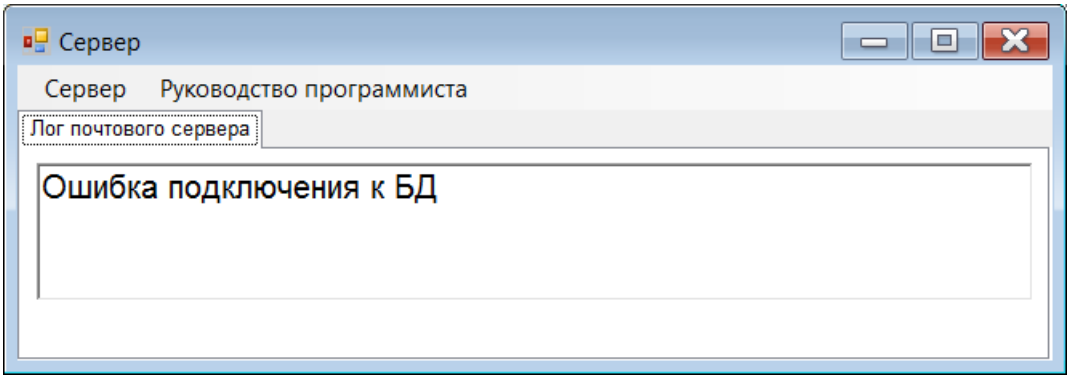

Рис. 4.7.1 – Вигляд у випадку неуспішного підключення до БД

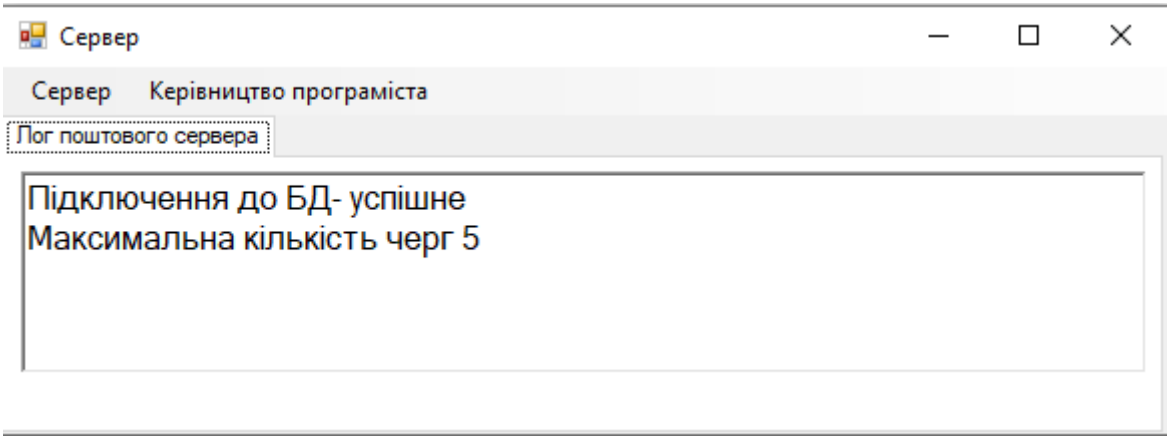

Рис. 4.7.2 – Вид додатку у випадку успішного підключення до БД

Даний проект складається з трьох модулів:

DB_Connection.cs – модуль підключення до бази даних

Form_Main.cs – головна форма

Server.cs – модуль призначений для роботи з протоколом TCP в .NET.

Протокол TCP (Transmission Control Protocol гарантує доставку повідомлень і широко використовується в різних існуючих насьогодні програмах.

Для роботи з протоколом TCP в .NET призначені класи TcpClient і TcpListener. Ці класи будуються поверх класів System.Net.Sockets.Socket.

Для створення клієнтської програми, яка працює по протоколу TCP, призначений клас TcpClient.

Для підключення до серверу TCP, в цьому класі визначений метод Connect(), якому передається назва хосту і порту:

```
TcpClient tcpClient = new TcpClient ();
tcpClient.Connect ("www.microsoft.com", 80);
```

Щоб взаємодіяти з сервером TcpClient визначається метод GetStream(), який повертає об’єкт NetworkStream. Через даний об’єкт можна передавати повідомлення серверу або, навпаки, отримувати дані з сервера.

Наведемо лістинг даного модуля, тому що він аналогічний таким же модулем в додатку ClientApp.exe, QAManager.exe, QADisplay.exe.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
namespace ServerApp
{
    class Server
    {
        TcpListener Listener;
        //старт сервер
        public Server(int Port)
        {
            Listener = new TcpListener(IPAddress.Loopback, Port);
            Listener.Start();
        }
        //стоп сервер
        ~Server()
        {
            if(Listener != null)
            {
                Listener.Stop();
            }
        }

        public TcpClient AcceptTcpClient()
        {
            return Listener.AcceptTcpClient();
        }

        public bool Pending()
        {
            return Listener.Pending();
        }
        //получить сообщение
        public StringBuilder GetResponse(NetworkStream stream)
        {
            byte[] data = new byte[256];
            StringBuilder response = new StringBuilder();

            do
            {
                int bytes = stream.Read(data, 0, data.Length);
                response.Append(Encoding.Unicode.GetString(data, 0, bytes));
            } while (stream.DataAvailable);

            return response;
        }
        //оправка сообщения

        public void Send(NetworkStream stream, string message)
        {
            byte[] data = Encoding.Unicode.GetBytes(message);
            stream.Write(data, 0, data.Length);
```

```
        }
    }
}
```

При запуску головної форми виконується наступний код (запускається сервер):

```
        DB_Connection.DBConnect();
            richTextBox1.Clear();

            //создаем экземпляр сервера
            server = new Server( 8080);
            if (DB_Connection.countMaxQueue==0)
            { richTextBox1.Text += "Ошибка подключения к БД"; }
            else
            richTextBox1.Text += "Подключение к БД - успешно"+Environment.NewLine+"Максимальное
количество очередей "+DB_Connection.countMaxQueue.ToString();
```

А в компоненті timer обробляються запити від клієнтів і виконується методи класу DB_Connection:

```
    //каждую сработку timera проверяем входящие запроси от клиентов
        private void timer1_Tick(object sender, EventArgs e)
        {
            if (server.Pending() == true)
            {
                TcpClient client = server.AcceptTcpClient();

                NetworkStream stream = client.GetStream();
                //получаем сообщение, считиваем параметри в массив
                string[] param = server.GetResponse(stream).ToString().Split('~');

                ArrayList arr;

                string res = string.Empty;

                switch (param[1])
                {

                    //запрос на входящие от клиента: параметр - код сотрудника
                    case "Специальности":
                                                arr = DB_Connection.GetSpeciality();

                        if (DB_Connection.GetError() == string.Empty)
                        {
                            res = string.Empty; foreach (string s in arr) res += s + "|";
                            //отправляем
                            server.Send(stream, res);
                        }
                        else
                        {//отправляем ошибку
                            server.Send(stream, DB_Connection.GetError());
                        }
                        break;
//запрос на факультети
```

```
case "Факультети":

    arr = DB_Connection.GetFaculties();

    if (DB_Connection.GetError() == string.Empty)
    {
        res = string.Empty; foreach (string s in arr) res += s + "|";
        //отправляем
        server.Send(stream, res);
    }
    else
    {//отправляем ошибку
        server.Send(stream, DB_Connection.GetError());
    }
    break;
case "детали":
                            arr = DB_Connection.GetDetali();

    if (DB_Connection.GetError() == string.Empty)
    {
        res = string.Empty; foreach (string s in arr) res += s + "|";
        //отправляем
        server.Send(stream, res);
    }
    else
    {//отправляем ошибку
        server.Send(stream, DB_Connection.GetError());
    }
    break;

case "Столи":

    arr = DB_Connection.getTableWTable();

    if (DB_Connection.GetError() == string.Empty)
    {
        res = string.Empty; foreach (string s in arr) res += s + "|";

        server.Send(stream, res);
    }
    else
    {//отправляем ошибку
        server.Send(stream, DB_Connection.GetError());
    }
    break;
case "колтекочередь":

    arr = DB_Connection.GetCountQueueCurrent();

    if (DB_Connection.GetError() == string.Empty)
    {
        res = string.Empty; foreach (string s in arr) res += s + "|";

        server.Send(stream, res);
    }
    else
    {//отправляем ошибку
        server.Send(stream, DB_Connection.GetError());
    }
    break;
```

```
                    case "текочередь":

                        arr = DB_Connection.GetCurrentQueue();

                        if (DB_Connection.GetError() == string.Empty)
                        {
                            res = string.Empty; foreach (string s in arr) res += s + "|";

                            server.Send(stream, res);
                        }
                        else
                        {//отправляем ошибку
                            server.Send(stream, DB_Connection.GetError());
                        }
                        break;
                    case "текочередьдлядисплея":

                        arr = DB_Connection.GetCurrentQueueForDisplay();

                        if (DB_Connection.GetError() == string.Empty)
                        {
                            res = string.Empty; foreach (string s in arr) res += s + "|";

                            server.Send(stream, res);
                        }
                        else
                        {//отправляем ошибку
                            server.Send(stream, DB_Connection.GetError());
                        }
                        break;
                    case "getnumber":


                        if (DB_Connection.GetError() == string.Empty)
                        {
                            res = string.Empty; res =
DB_Connection.InsertClientToQueue_GetNamber(param[0].ToString(), param[2].ToString(),
param[3].ToString(), param[4].ToString(),param[5].ToString());

                            server.Send(stream, res);
                        }
                        else
                        {
                            server.Send(stream, DB_Connection.GetError());
                        }
                        break;
                    case "setintstatus0":


                        if (DB_Connection.GetError() == string.Empty)
                        {

                            res = string.Empty; res =
DB_Connection.setintstatus0(param[0].ToString(), param[2].ToString(), param[3].ToString(),
param[4].ToString(), param[5].ToString());

                            server.Send(stream, res);
                        }
                        else
                        {
```

```
                        server.Send(stream, DB_Connection.GetError());
                    }
                    break;
                case "serverisrunning":


                    if (DB_Connection.GetError() == string.Empty)
                    {
                        res = string.Empty; res =
DB_Connection.GetServerIsRunning(param[0].ToString());

                        server.Send(stream, res);
                    }
                    else
                    {
                        server.Send(stream, DB_Connection.GetError());
                    }
                    break;



//запрос на користувача
                case "Польз":
                    richTextBox1.Text += "Запрос на виборку пользователей " +
Environment.NewLine;
                    arr = DB_Connection.GetUsers();
                    res = string.Empty; foreach (string s in arr) res += s + "~";

                    server.Send(stream, DB_Connection.GetError() == string.Empty ? res :
DB_Connection.GetError());
                    break;
                    //запрос на получение таблици сотрудников
                case "ВсеПольз":
                    richTextBox1.Text += "Запрос на виборку расширенной информации о
користувачах " + Environment.NewLine;
                    arr = DB_Connection.GetUsersAll();
                    res = string.Empty; foreach (string s in arr) res += s + "|";

                    server.Send(stream, DB_Connection.GetError() == string.Empty ? res :
DB_Connection.GetError());
                    break;
//запрос на вход
                case "Вход":
                    richTextBox1.Text += "Вход користувача " + param[0] + " " + param[2] +
Environment.NewLine;

                    if (DB_Connection.FindUser(param[0], param[2]) == true)
server.Send(stream, "1");
                    else server.Send(stream, "0");
                    break;
//запрос на получение кода сотрудника
                case "Я":
                    server.Send(stream, DB_Connection.GetUserCode(param[0]).ToString());
                    break;
                case "fio":
                    server.Send(stream, DB_Connection.GetFullName(param[0]).ToString());
                    break;


            }
```

```
            //закриваем поток
            stream.Close();
            //закриваем клиента
            client.Close();
        }
           }
```

В модулі DB_Connection.cs здійснюється підключення до бази даних і формується звіти на запити клієнтів. Детальне підключення до БД описано далі.

```
static public void DBConnect()
        {
            try
            {
                string pathfile = Directory.GetCurrentDirectory() + "\\Path.inf";
                if (File.Exists(pathfile))
                {
                    try
                    {
                        FileStream fs = new FileStream(pathfile, FileMode.Open);
                        StreamReader sr = new StreamReader(fs, Encoding.UTF8);
                        connectstring = sr.ReadLine();
                        sr.Close();
                    }
                    catch (Exception ex)
                    {
                        connectstring = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" + dbPath +
dbName + @";Integrated Security=True;Connect Timeout=30";
                    }
                }
                else
                {
                    connectstring = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" + dbPath +
dbName + @";Integrated Security=True;Connect Timeout=30";
                }

                dbError = string.Empty;
                string connectionString = connectstring;
                con = new SqlConnection(connectionString);
                dbQuery = "SELECT countMaxQueue  from settingsQueue where idSetting=1";
                dBcom = new SqlCommand(dbQuery, con);
                con.Open();
                countMaxQueue = (Convert.ToInt32(dBcom.ExecuteScalar()));
                con.Close();
            }
            catch (SqlException ex)
            {
                dbError = "Невозможно подключиться к базе даних! " + ex.ToString();
            }
            finally
            {
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }
               }
```

Приклад оброблення запитів на зміну статусів черги приведений нижче:

```
      static public string setintstatus0(string idUser, string Number, string intStatus, string
idTable, string time)
        {
            try
            {
                string sql = "";
                if (intStatus == "1")
                {
                     sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timeCall=@time
where Number=@Number and DATEDIFF(day, timeRegistration, GETDATE()) = 0";
                }
                if (intStatus == "2")
                {
                     sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timeServiceS=@time
where Number=@Number  and DATEDIFF(day, timeRegistration, GETDATE()) = 0;";
                }
                if (intStatus == "11" || intStatus == "12" || intStatus == "13" || intStatus ==
"14" || intStatus == "15" || intStatus == "16" || intStatus == "17")
                {
                    sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timePutOff=@time
where Number=@Number  and DATEDIFF(day, timeRegistration, GETDATE()) = 0;";
                }
                if (intStatus == "9" || intStatus == "10" || intStatus == "17")
                {
                    sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timeEndService=@time
where Number=@Number  and DATEDIFF(day, timeRegistration, GETDATE()) = 0;";
                }
dBcom = new SqlCommand(sql, con);
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@idUser", DbType =
System.Data.DbType.Int32, Value = Convert.ToInt32(idUser) });
if (intStatus == "17")
    dBcom.Parameters.Add(new SqlParameter { ParameterName = "@intStatus", DbType =
System.Data.DbType.Int32, Value = 10 }); else dBcom.Parameters.Add(new SqlParameter {
ParameterName = "@intStatus", DbType = System.Data.DbType.Int32, Value =
Convert.ToInt32(intStatus) });
if (idTable == "0") dBcom.Parameters.Add(new SqlParameter { ParameterName = "@idTable", DbType =
System.Data.DbType.Int32, Value = DBNull.Value});
else dBcom.Parameters.Add(new SqlParameter { ParameterName = "@idTable", DbType =
System.Data.DbType.Int32, Value = Convert.ToInt32(idTable) });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@time", DbType =
System.Data.DbType.DateTime, Value = Convert.ToDateTime(time) });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@Number", DbType =
System.Data.DbType.Int32, Value = Convert.ToInt32(Number)});
con.Open();
```

```
dBcom.ExecuteScalar();
con.Close();
                        return ("1");

            }
            catch (SqlException ex)
            {
               return "Ошибка" + ex.ToString();
            }
     finally
            {
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }
        }
```

4.7.1 Технологія підключення до бази даних.

Для створення СУБД-додатку необхідно підключитися до джерела даних, щоб мати можливість керувати їм. В об'єктній моделі ADO.NET з'єднання з вашим джерелом даних забезпечує Connection. Об'єкти Connection також є відправною точкою для створення запитів і транзакцій.

Об'єкт ADO.NET Connection в залежності від обраного джерела даних цей об'єкт може називатися по-різному. Для створення підключення до баз даних MS SQL Server треба використовувати Об'єкт SqlConnection, який знаходиться в просторі імен System.Data.SqlClient

Простір імен System.Data.SqlClient є постачальником даних .NET FrameWork для источников даних MS SQL Server.

Для роботи з об'єктом SqlConnection йому потрібно надати рядок з'єднання, який вказує, яким чином потрібно підключитися до джерела даних. Рядок з'єднання — рядок, який складається з пар імен — значення, що містить відомості про ініціалізації, передавати у вигляді параметра від додатка до джерела даних. Синтаксис рядка з'єднання залежить від обраного джерела даних.

Рядок підключення:

Формат рядка з’єднання є списком, розділені крапкою з комою пар «параметр-значення». Знак рівності (=) з’єднує кожний параметр з його значенням.

Основні параметри рядка підключення до MS SQL Server БД:

Data Source – вказує ім’я екземпляра SQL Server, до якого потрібно підключитися.

Initial Catalog – параметр, який вказує на ім’я бази даних на сервері, до якому потрібно підключитися.

Integrated Security – дозволяє використовувати для підключення до серверу данні обліково запису Windows або ім’я входу SQL Server.

User Id — дозволяє вказати ім’я SQL Server для підключення до серверу

Password — пароль імені входа SQL Server

Варіанти підключення:

string conStr = @"Data Source=.\SQLEXPRESS;Initial Catalog=MyDB; Integrated Security=True";

string conStr = @"Data Source=(local)\SQLEXPRESS;Initial Catalog=MyDB; Integrated Security=True";

string conStr = @"Data Source=localhost\SQLEXPRESS;Initial Catalog=MyDB; Integrated Security=True";

Вищеприведених прикладах підключення можно зчитати наступним чином:

Дотримуючись інструкції, рядки підключення потрібно знайти на локальному комп’ютері екземпляр SQL Server з іменем SQLEXPRESS, пошукати каталог MyDB і спробувати отримати доступ до джерела даних через довірене підключення, використовуючи для цього ваш обліковий запис Microsoft Windows

В нашому випадку рядок підключення буде виглядати так:

Data Source=.\SQLEXPRESS; AttachDbFilename = d:\DirQBase\QBase.mdf;Integrated Security=True;Connect Timeout=30.

Даний рядок запишемо в файл Path.inf і в додатку організуємо зчитування даного рядка і запис у властивість Connection.

4.8 Додаток AdminApp.exe

Додаток AdminApp.exe призначений для редагування довідника бази даних і налаштування шляху максимальної кількості черги. Даний додаток безпосередньо взаємодіє з базою даних Qbase.mdf.

Даний проект складається з наступних модулів:

ClassQBase.cs – клас підключення до бази даних.

Client.cs – модуль призначений для роботи з протоколом TCP в .NET.

fdetailSpecialty.cs – форма вид спеціальності;

Form1.cs – головна форма;

fabout.cs – форма про програму;

FormChangeParol.cs – форма зміни паролю.

FormSpr.cs – форма редагування простих довідників, які складаються з 1 поля.

fsotrudnik.cs – форма редагування даних співробітника.

fspeciality.cs - форма редагування спеціальності.

fspecialtyontable.cs – форма редагування прив’язки до робочого столу.

fwtable.cs – форма редагування робочого столу.

LoginForm.cs – форма аутентифікації.

Program.cs – програмний модуль. У ньому розміщені деякі глобальні змінні.

Основні функції даного додатку є редагування таблиць БД (крім самої електронної черги). Підключення до бази даних здійснюється аналогічно підключення додатку ServerApp.exe. Приклад роботи додатку приведено на рис. 9 – 4.8.23.

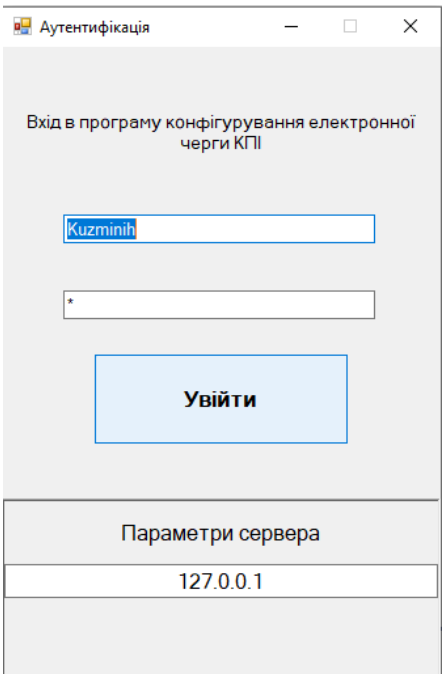

Рис. 10 – Форма аутентифікації

При відключеному сервері можливо переглянути конфігурації системи, перегляд і друк статистичних даних. Також можна змінювати прив'язку обслуговування черг до робочих місць і налаштовувати довідники.

Якщо програма сервера активна, програма конфігурування запускається в режимі тільки для зчитування. Для виходу з даного режиму необхідно зупинити сервер системи й перезапустити програму конфігурації. Дана ситуація виникає практично завжди при віддаленому перегляді статистики.

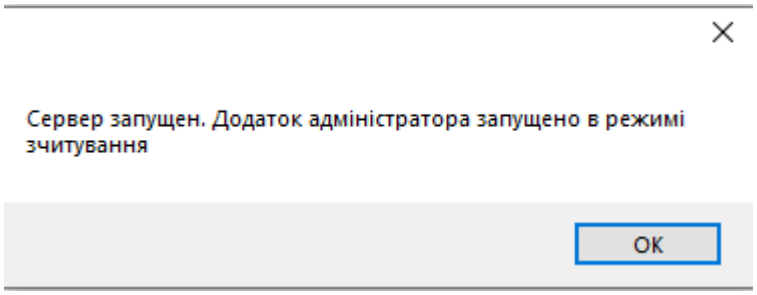

Рис.11 – Попередження про режим зчитування

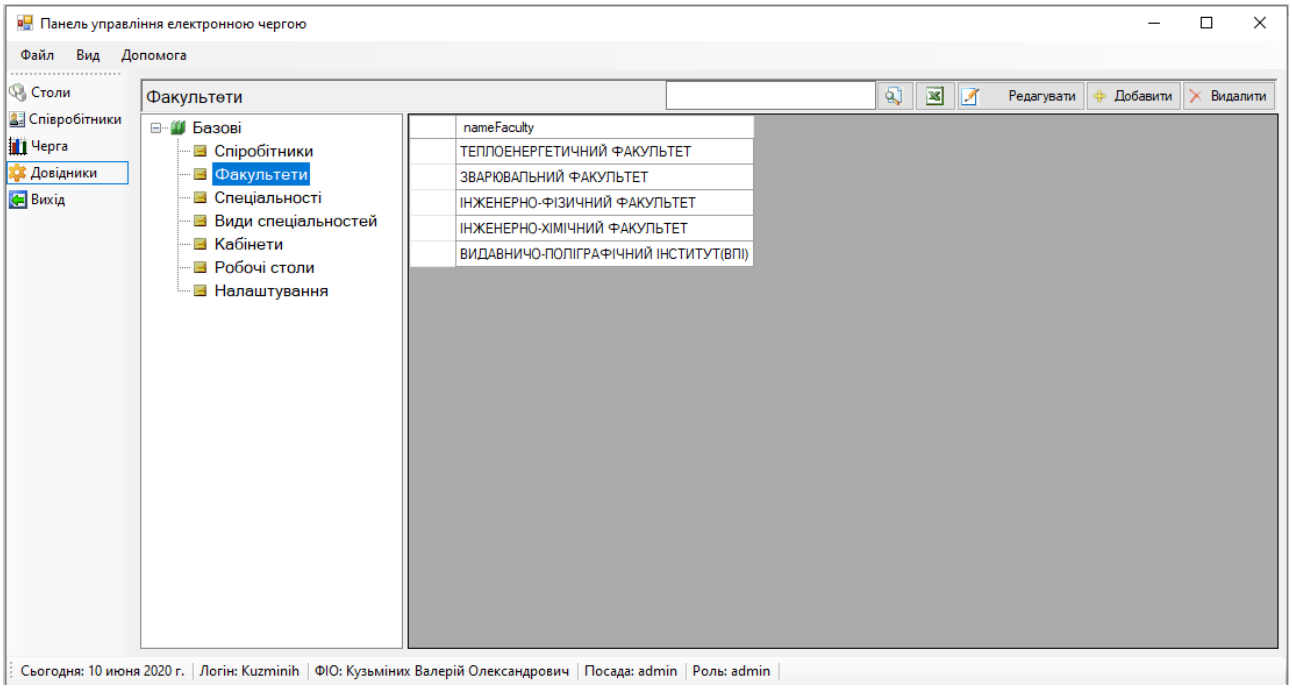

Рис.12 – Довідник Факультети

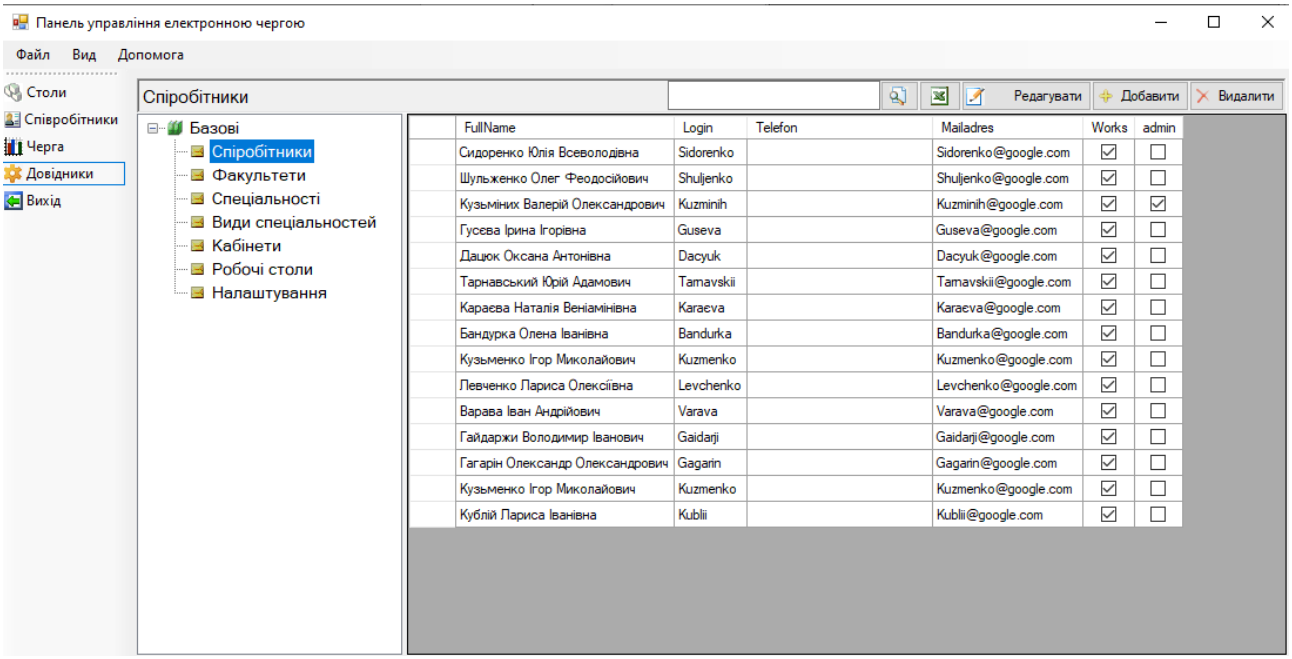

Рис.13 – Довідник співробітники

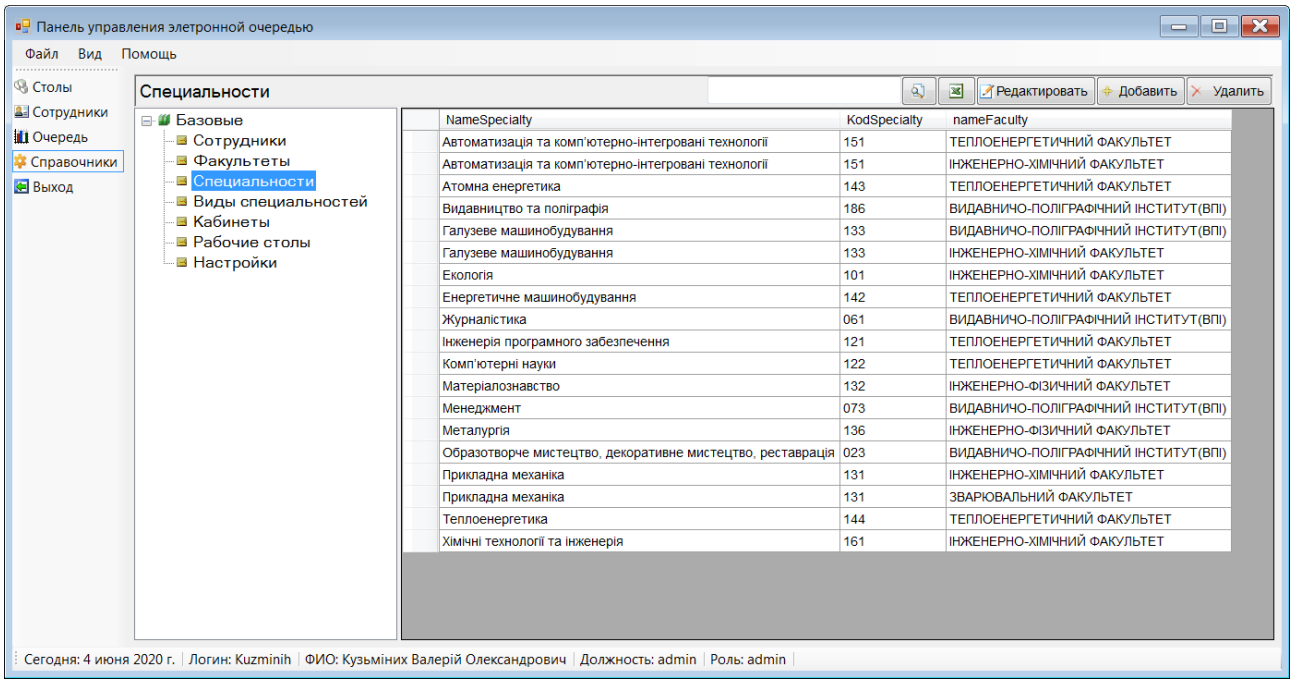

Рис.14 – Довідник Довідники

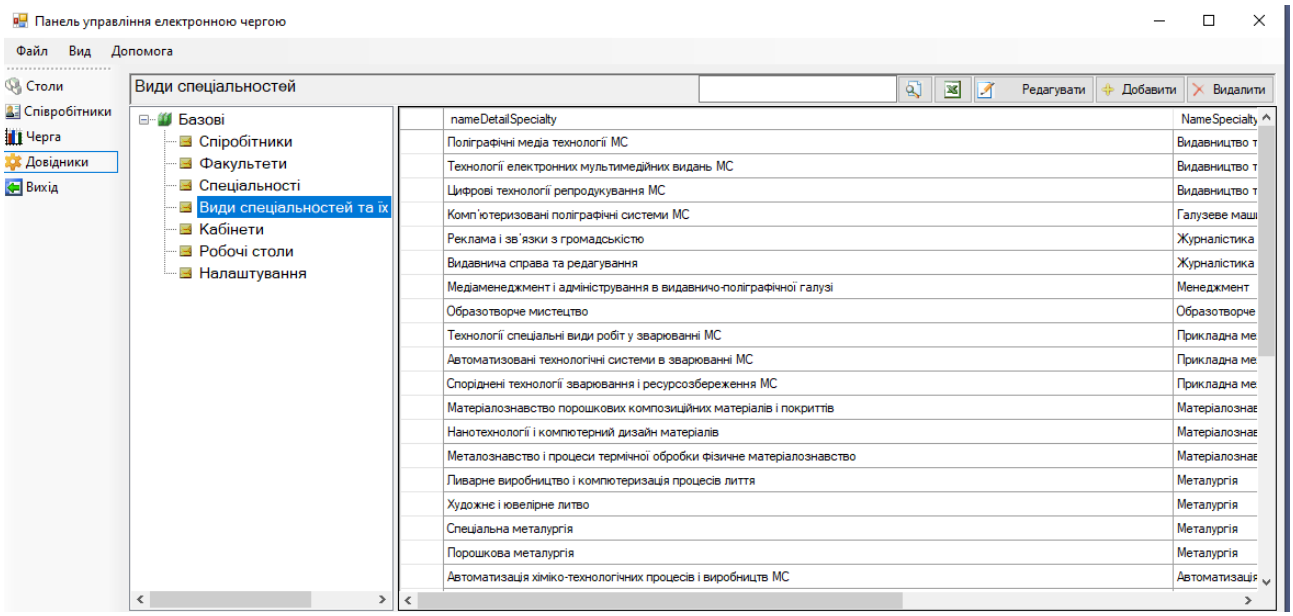

Рис.4.8.6 – Довідник види спеціальностей та їх спеціалізації

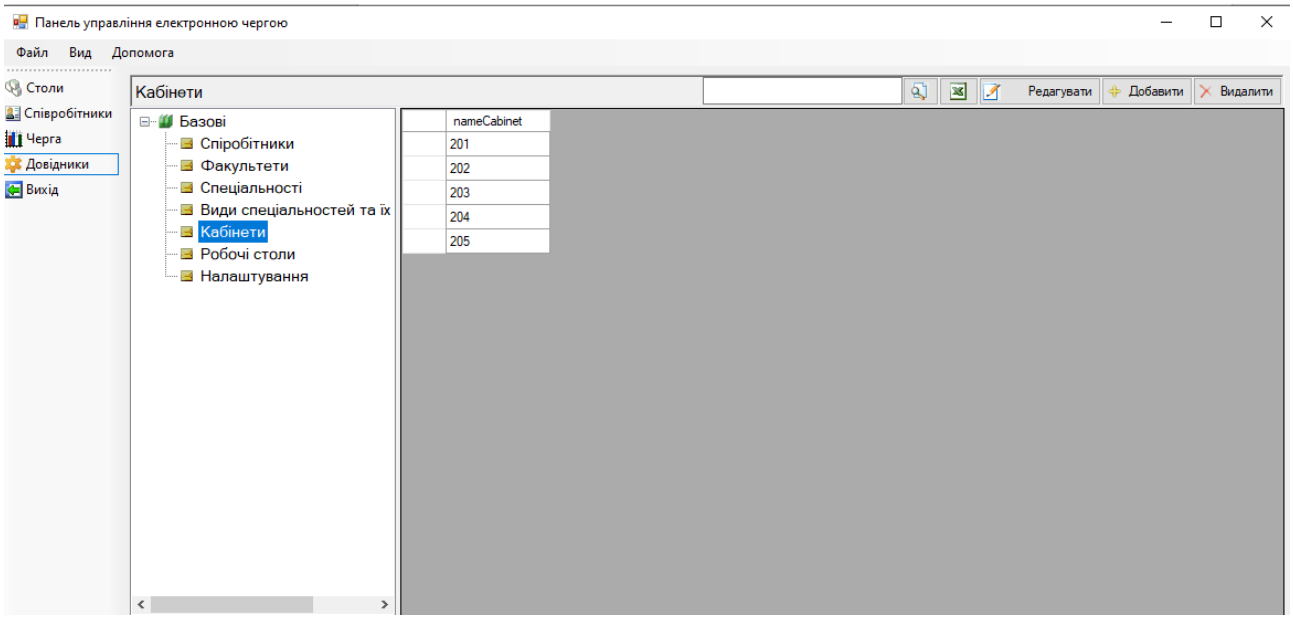

Рис.15 – Довідник Кабінетів

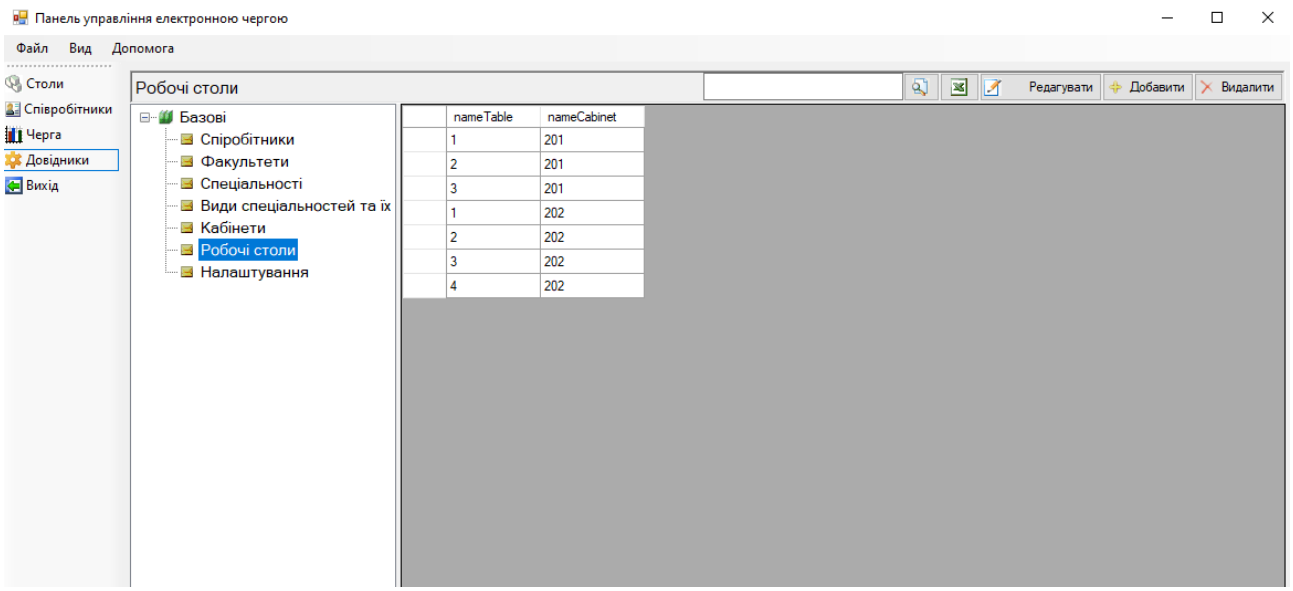

Рис.16 – Довідник Робочі столи

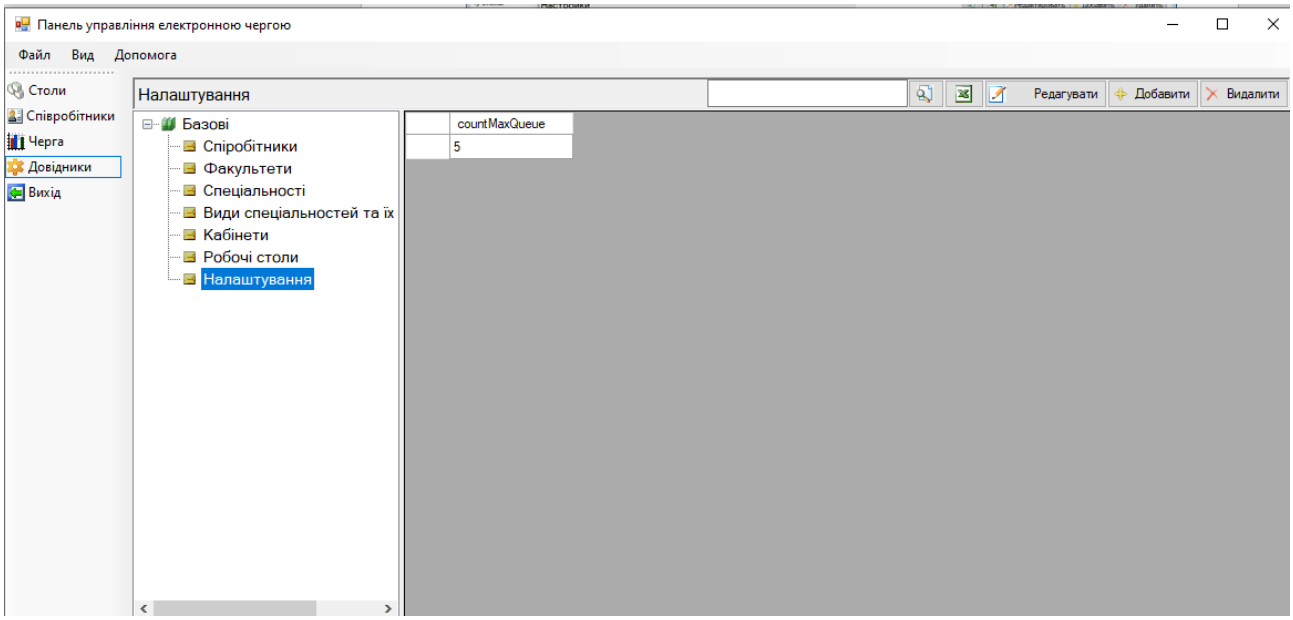

Рис.17 – Довідник Налаштування

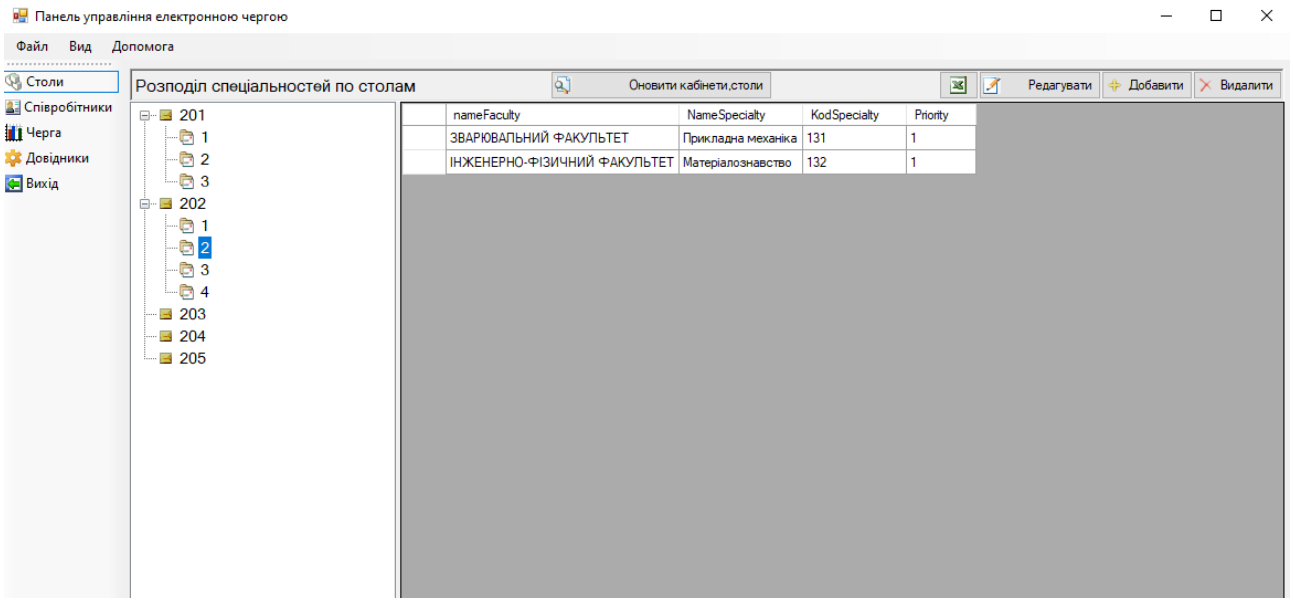

Рис.4.8.10 – Вкладка розподіл спеціальностей по столах

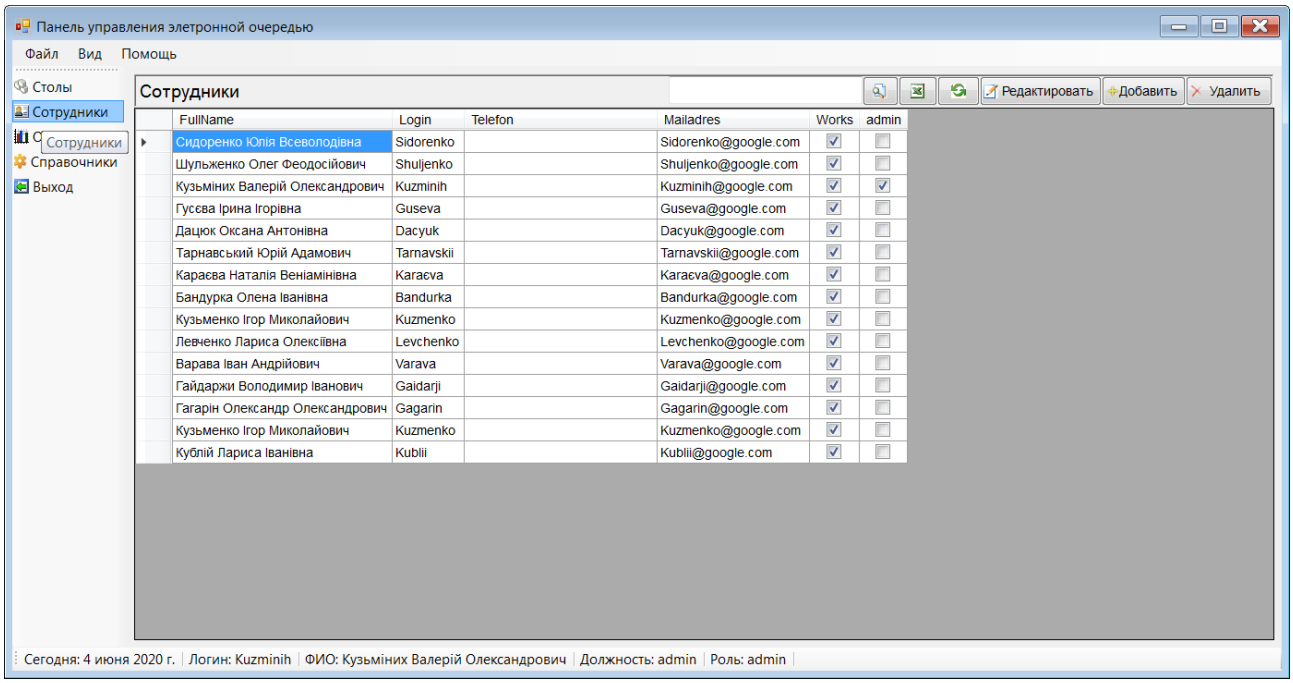

Рис.18 – Вкладка співробітники

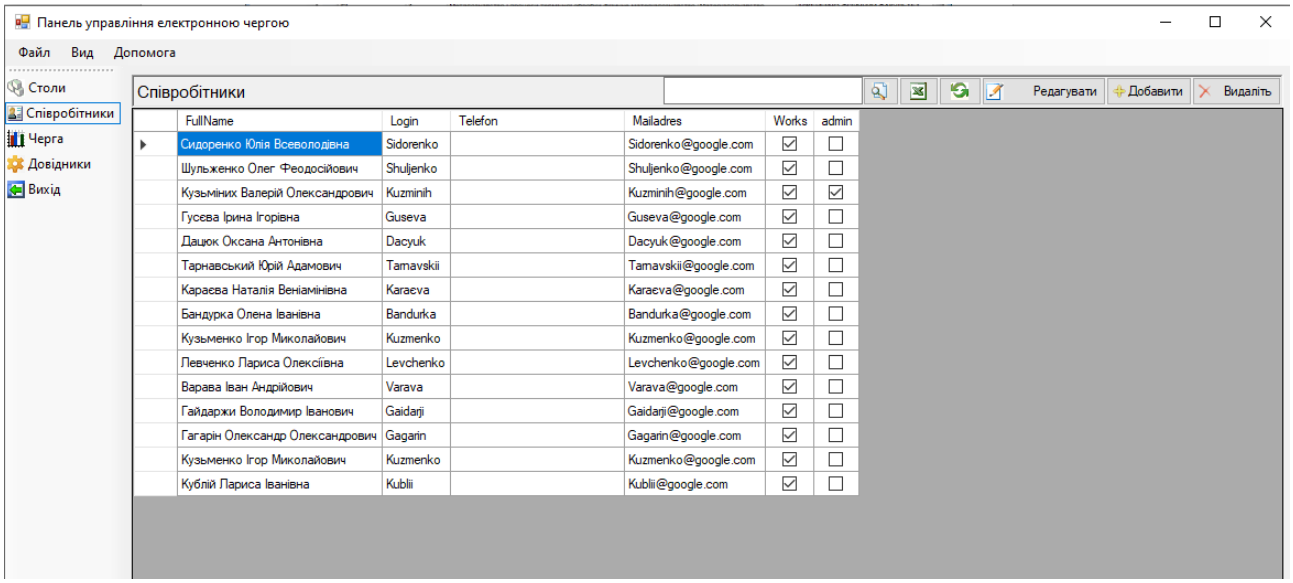

Рис.4.8.12 – Вкладка черга

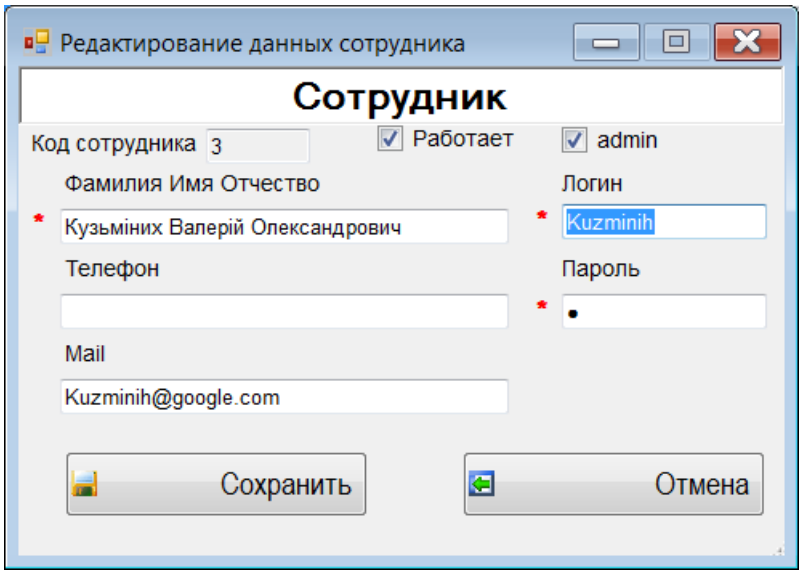

Рис.4.8.13 – Форма редагування/добавлення співробітник

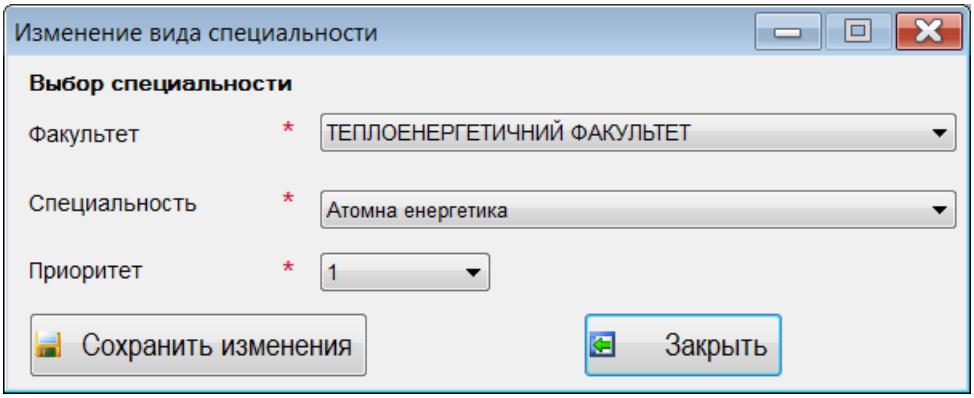

Рис.4.8.14 – Форма редагування/добавлення видів спеціальності

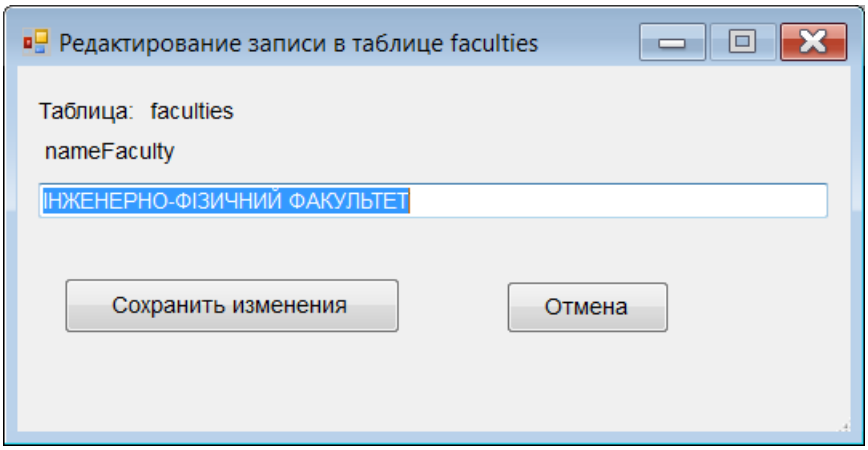

Рис.4.8.15 – Форма редагування/добавлення Факультету

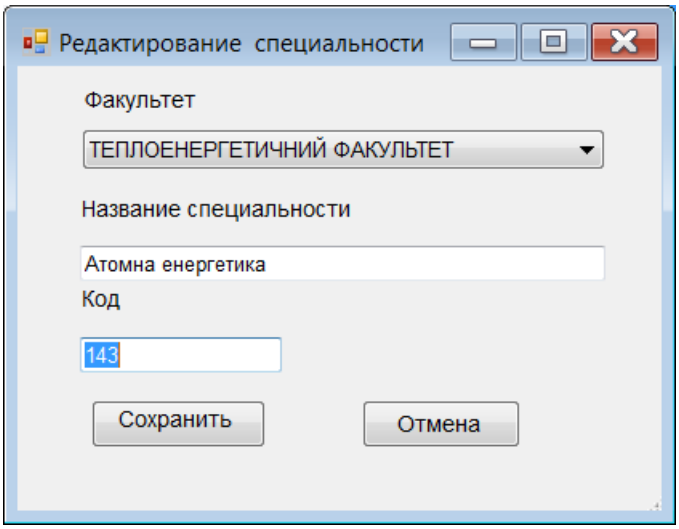

Рис.4.8.16 – Форма редагування/добавлення спеціальності

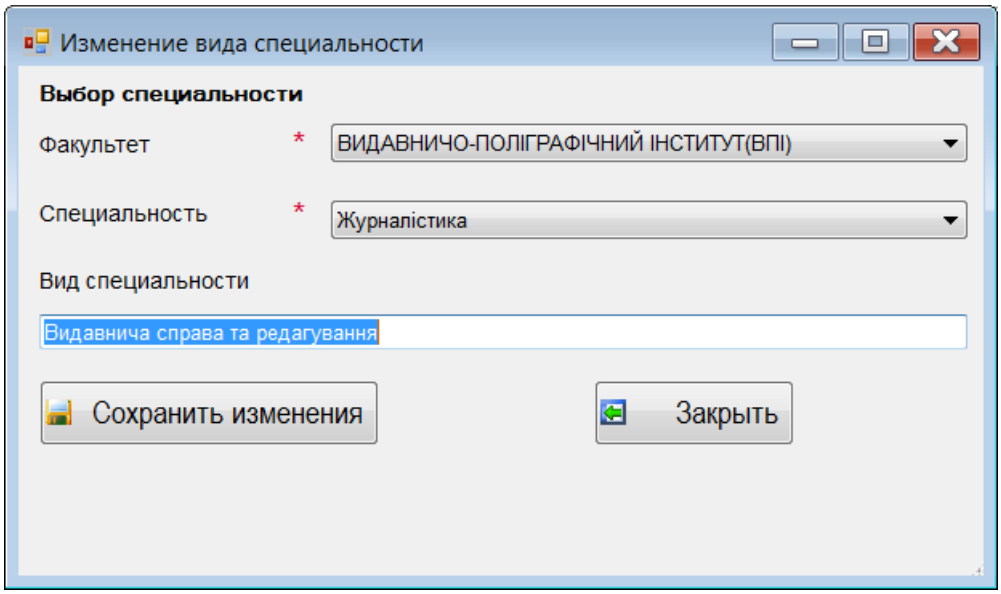

Рис.4.8.17 – Форма редагування/добавлення видів спеціальностей

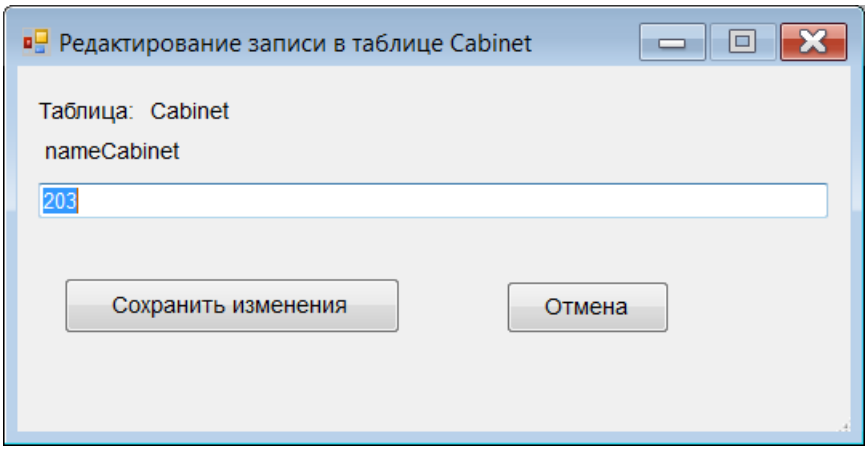

Рис.4.8.18 – Форма редагування/добавлення кабінету

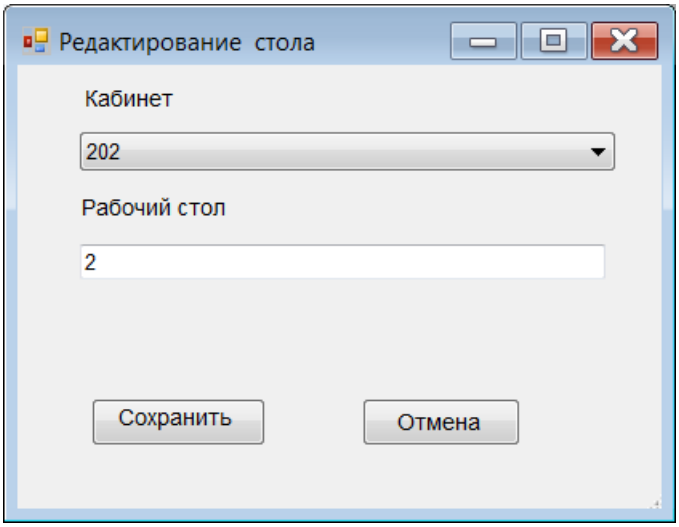

Рис.4.8.19 – Форма редагування/добавлення робочого стола

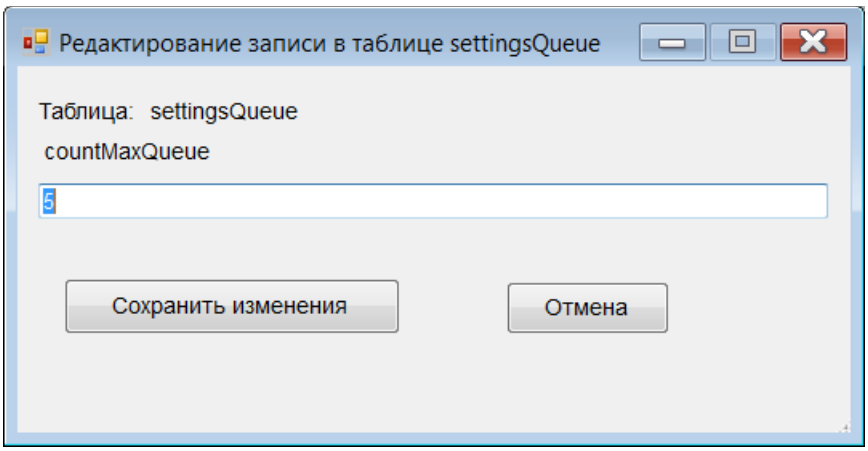

Рис.19 – Форма редагування значення максимальної черги

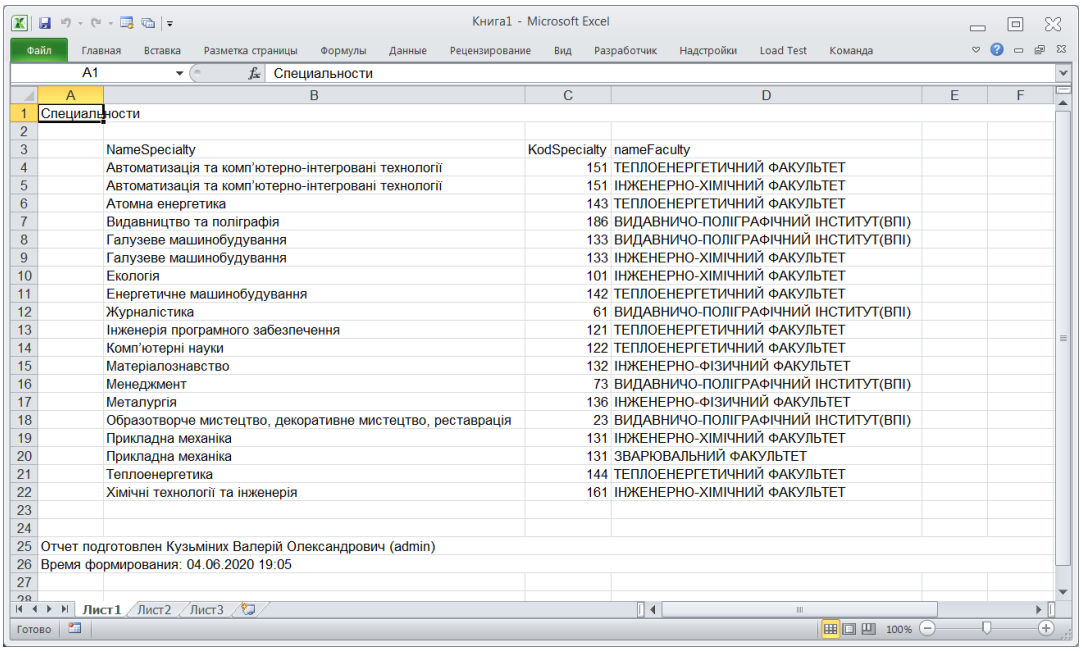

Рис.4.8.21 – Результат експорту в MS Excel

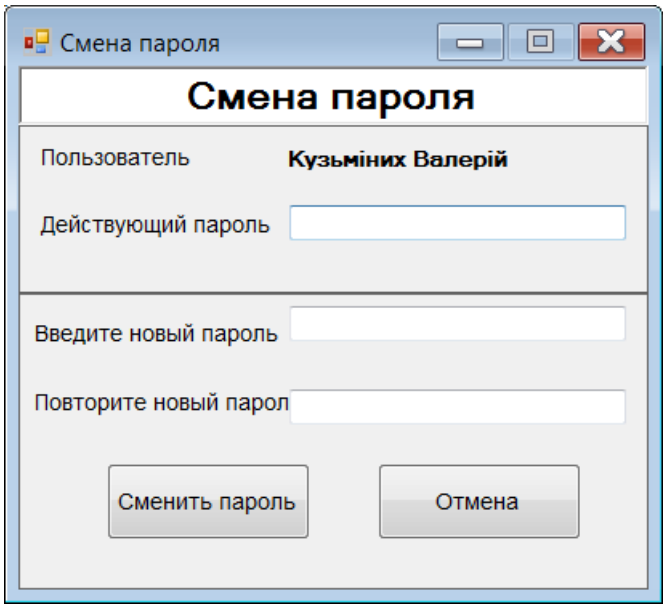

Рис.4.8.22 – Форма зміни паролю користувача

В разі спроби редагування, видалення або добавлення запитів запуску додатку, при запущеному серверу буде виведено повідомлення (рис.33):

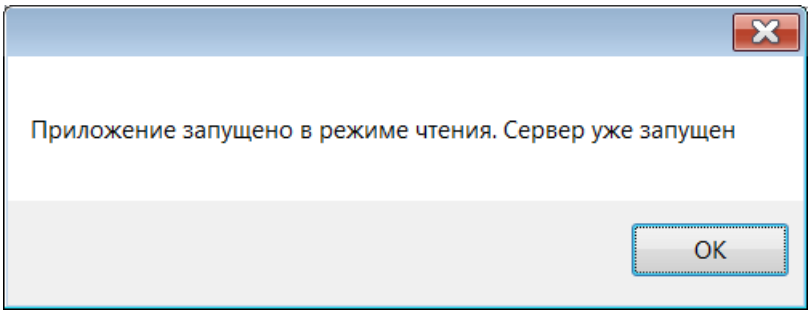

Рис.4.8.23 – Попереджуюче повідомлення

4.9 Додаток QAManager.exe

В наступних трьох додатках використовується клієнтська частина на основі класу TcpClient. До складу проектів ClientApp, QADisplay, QAManager входить модуль Client.cs, який відповідає за взаємозв'язок з сервером (ServerApp.exe). Приведемо лістинг даного модуля повністю, так як він дуже важливий і практично однаковий для всіх клієнтський додатків.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Windows.Forms;
namespace Qmanager
{
    //главний класс
    public class Client
    {
     //   private TcpClient client;
        private string server;
        private int port;
        public int idUser;
        public string userlogin;
        public string PathFileBase;
        public bool YesPathFileBase;
        public string mistake="";
      public int GetUser() { return idUser; }
        //инициализация client
        public Client(string server, int port, int idUser = -1)
        {
            this.server = server;
            this.port = port;
```

```
    this.idUser = idUser;
}
//меняем сервер
public void changeclientserver(string server)
{
    this.server = server;

}
//устанавливаем usera
public void SetUser(string Name)
{
    this.idUser = Convert.ToInt32(Send(Name + "~Я").ToString());
    userlogin = Name;
}

//стандартная операция отправления на сервер сообщения
public StringBuilder Send(string message)
{
    bool flag = false;
    TcpClient client = new TcpClient(); //новий экземпляр
    StringBuilder response = new StringBuilder();
    try
    {
        try
        {
            client.Connect(server, port);//коннектимся
        }
        catch (Exception ee)
        {
            MessageBox.Show("Ошибка подключения");
            response.Append("Ошибка");
            mistake = "Ошибка";
            return response;
        }
        NetworkStream stream = client.GetStream();//откриваем новий поток
        byte[] data = Encoding.Unicode.GetBytes(message);//преобразуем сообщение
        stream.Write(data, 0, data.Length);
        data = new byte[256];
        //отправляем
        do
        {
            int bytes = stream.Read(data, 0, data.Length);
            response.Append(Encoding.Unicode.GetString(data, 0, bytes));
        } while (stream.DataAvailable);
        //закриваем поток
        stream.Close();
        client.Close();//закриваем клиента
        flag = true;

    }
    catch (SyntaxErrorException ee)
    {
        mistake = "Ошибка";
       // MessageBox.Show("Ошибка подключения "+ee.ToString());
    }
    if (!flag)
    {
        response.Append("Ошибка");
        mistake = "Ошибка";
    }
```

```
                return response;  //полученний от сервера ответ
        }

        public ArrayList GetTable(string MailType)
        {
            //отправляем на сервер тип
            StringBuilder result = Send(idUser.ToString() + "~" + MailType);
            ArrayList arr = new ArrayList();
            //получаем от сервера массив
            arr.AddRange(result.ToString().Split('|'));
            return arr;
        }


//регистрация заявки
        public string GetNamber( string idSpecialityS, string idDeteilSpecialityS ,string
timeRegistrationS)
        {
            return Send(idUser.ToString() + "~getnumber~" + idSpecialityS + "~" +
idDeteilSpecialityS +"~"+timeRegistrationS
                + "~" + Program.nameClient + "|" + Program.dateBithClient + "|" + Program.FDP +
"|" + Program.MAN + "|" + Program.VUO + "|" + Program.OLIMP).ToString();
        }
        public string GetFullName()
        {
            return Send(idUser+ "~fio").ToString();
        }


        //ищем usera по логину и паролю
        public bool FindUser(string Name, string Parol)
        {
            if (mistake == "Ошибка") return false;
            return Send(Name + "~Вход~" + Parol).ToString() == "0" ? false : true;
        }
        //получаем данние по idUser
        public ArrayList GetUsers()
        {
            StringBuilder result = Send("0~Польз");
            ArrayList arr = new ArrayList();
            arr.AddRange(result.ToString().Split('~'));
            return arr;
        }

    }
}
```

Додаток QManager.exe призначений для реєстрації абітурієнта в електронній черзі. Даний додаток безпосередньо взаємодіє з додатком ServerApp посередництвом клієнта TcpClient.

Даний проект складається з наступних модулів:

Form1.cs – головна форма;

Form_Auth.cs – форма аутентифікації;

Form_Log.cs – форма відображення помилок програми;

Fabout.cs – про програму;

Client.cs class клієнта TcpClient;

Program.cs – програмний модуль. В ньому знаходиться деякі глобальні змінні.

Приклад роботи додатку приведені на рис. 4.9.1 - 4.9.5.

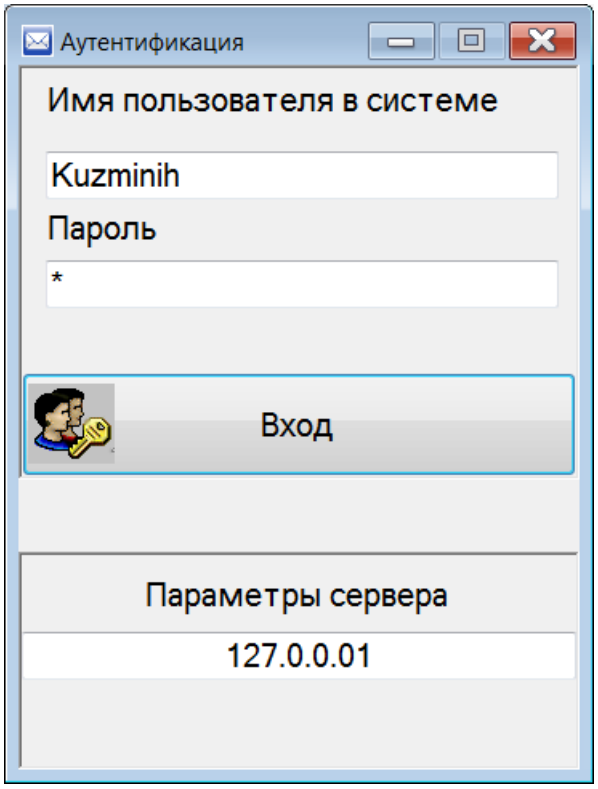

Рис.4.9.1 – Форма аутентифікації

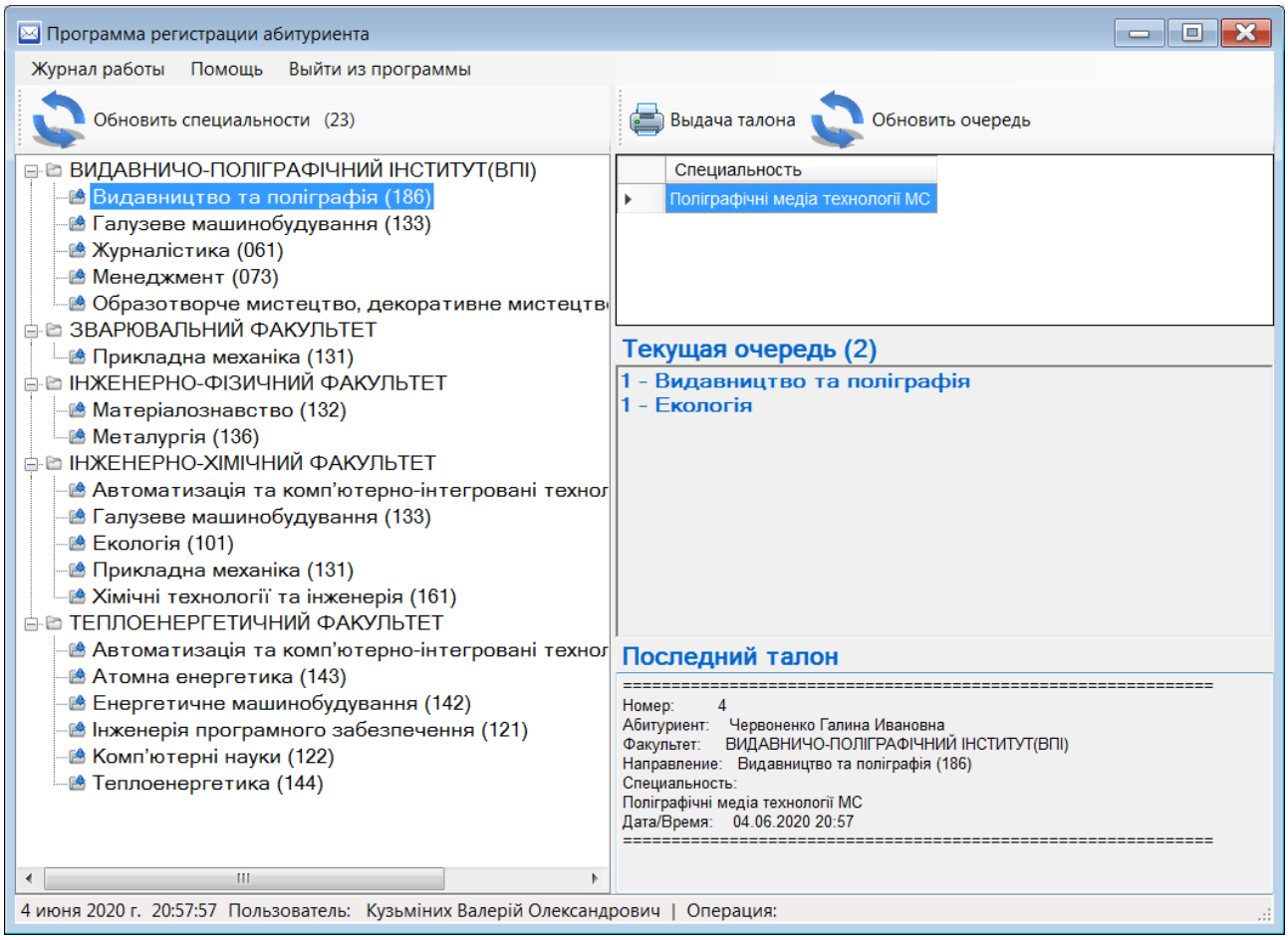

Рис.4.9.2 – Головна форма

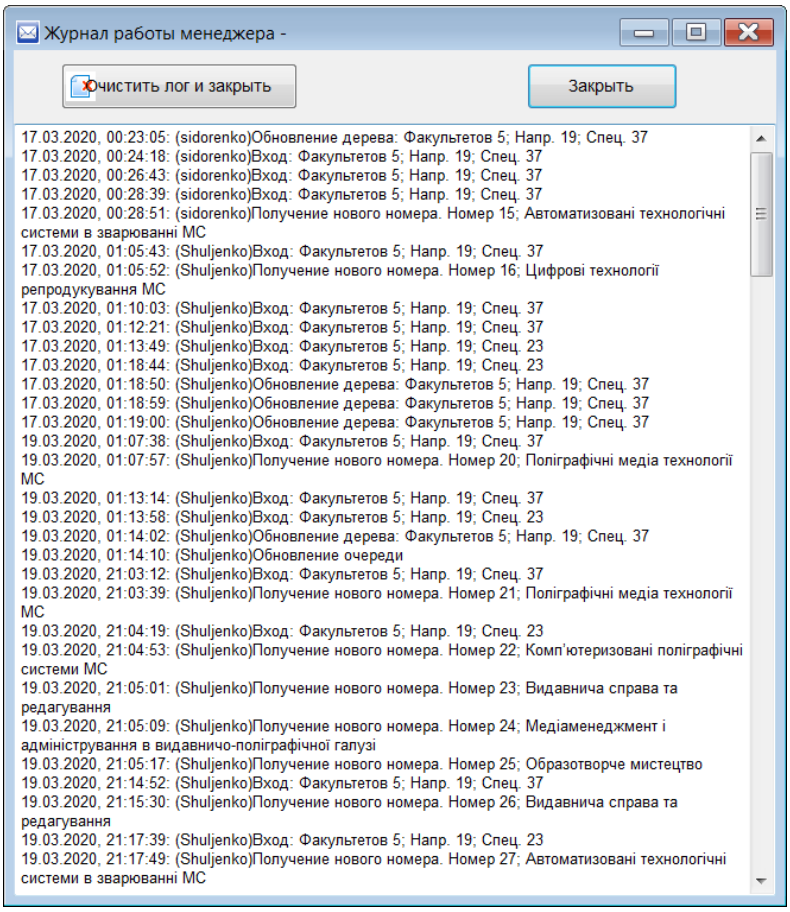

Рис.4.9.3 – Помилки(Лог) програми

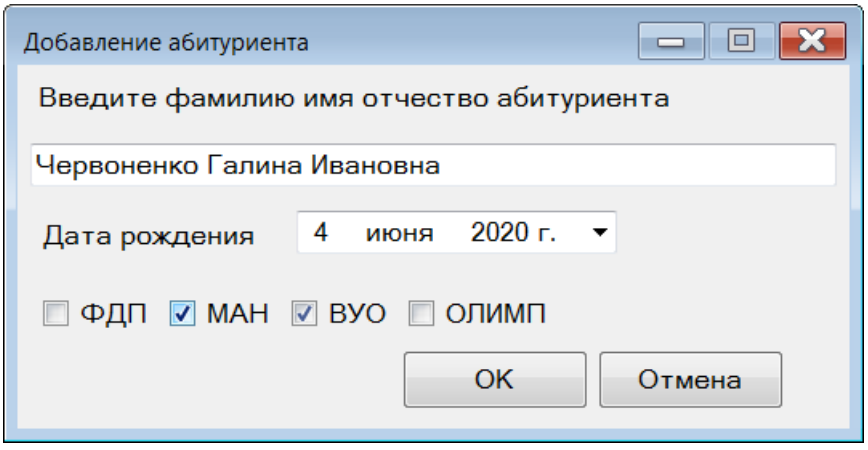

Рис.4.9.4 – Форма додавання абітурієнта

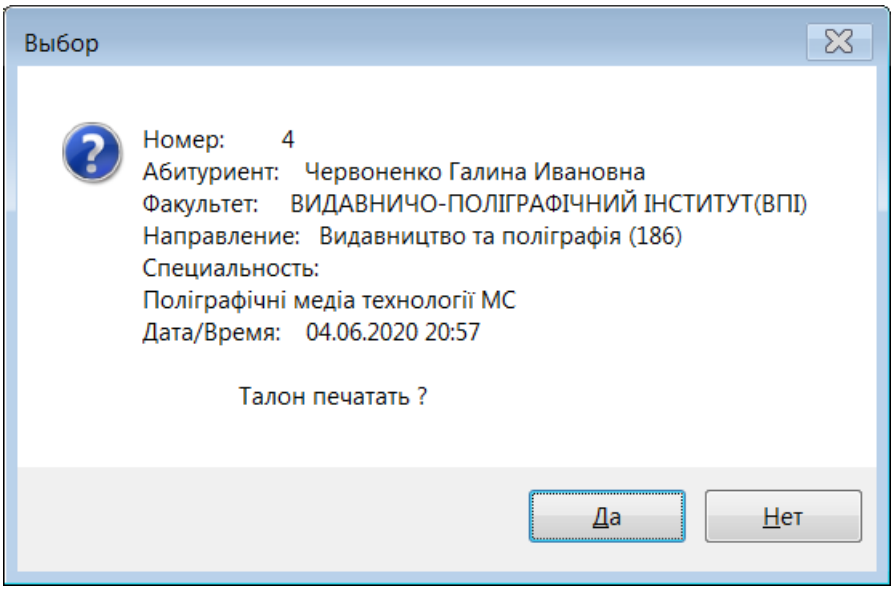

Рис. 4.9.5 – Інформація про абітурієнта

4.10 Додаток ClientApp.exe

Програма дозволяє:

- Автоматично з’єднатися з програмою сервера черг;
- Отримувати інформацію, скільки абітурієнтів чекають в чергах, для обслуговування яких призначено дане робоче місце;
- Запросити наступного абітурієнта до робочого місця для його подальшого обслуговування, використавши кнопку виклику;
- Підтвердити факт завершення обслуговування або неявки абітурієнта;
- Відкласти виклик абітурієнта на деякий час у випадку його неявки;

Даний проект складається з наступних модулів:

- Form_Client.cs – головна форма;
- Form_Auth.cs – форма аутентифікації;
- Client.cs class клієнта TcpClient;
- fdesktopSelection.cs – вікно вибору робочого столу при вході в програму;

Програма обслуговування абітурієнтів призначена для виклику абітурієнта і повинна бути запущена на комп’ютері кожного робочого місця. Програма може бути в одному з двох станів – підключена до серверу черги або відключена від нього.

При підключенні до серверу відкривається вікно вводу логіна користувача і пароля. Ввід даних обов’язковий. При вводі помилкових даних більш ніж 3 рази відбувається завершення роботи програми (рис. 39-44).

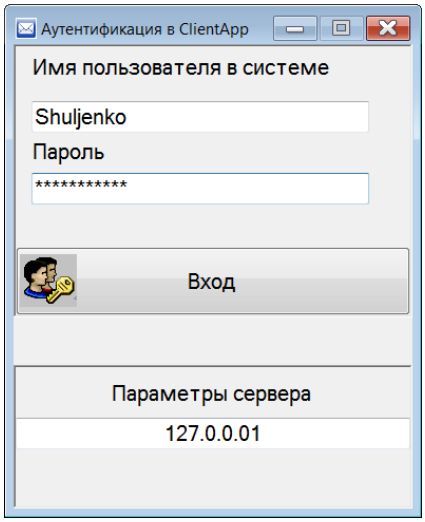

Рис. 4.10.1 – форма аутентифікації

Логін и ір адрес сервера зберігаються в файлі налаштувань server.ini. Після вдалої аутентифікації необхідно обрати робочий стіл.

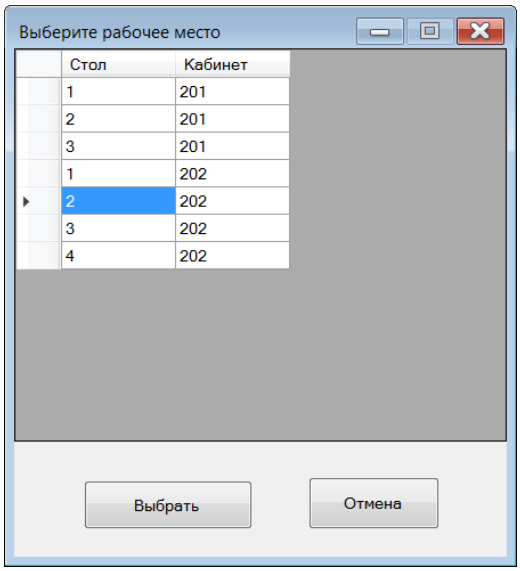

Рис. 4.10.2 – Форма вибору робочого стола

Потім відкривається вікно секретаря.

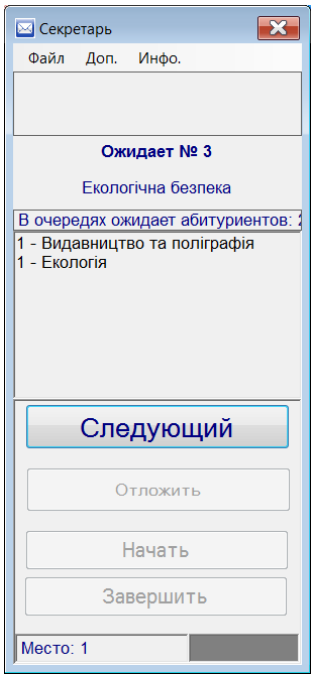

Рис. 4.10.3 – Форма секретаря

Якщо в черзі є абітурієнти, кнопка Наступний включена і ви бачите номер талона і назва черги чекаючого абітурієнта. Щоб викликати абітурієнта натисніть

кнопку Наступний. Додаткові кнопки дозволяють відкласти виклик абітурієнта у випадку неявки, викликати абітурієнта по номеру, підтвердити факт виклику або неявки абітурієнта і визначити параметри реєстрації білета (інформація про абітурієнта).

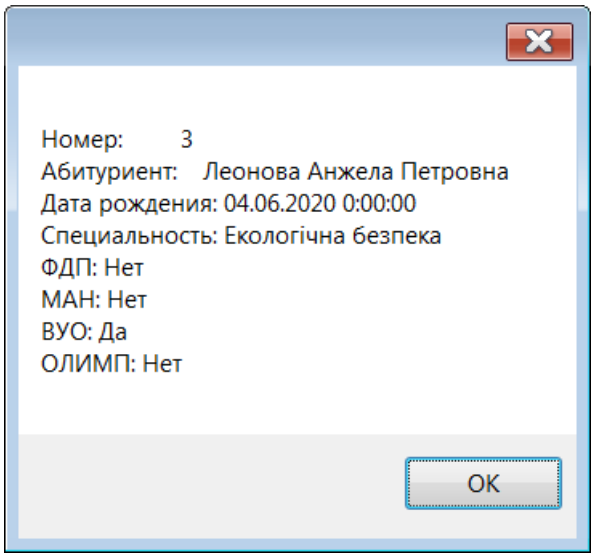

Рис. 4.10.4 – Інформація про абітурієнта

Конкретне призначення кнопки можна визначити по тексту на кнопці. Кнопки включаються і виключаються автоматично, в залежності від поточного стану програми. При натисканні кнопки наступний факт обслуговування попереднього користувача підтверджується автоматично. Про стан програми можна дізнатися по кольору піктограми в панелі задач або по кольору прямокутника в рядку стану:

- Сірий – в черзі немає абітурієнтів;
- Синій – в черзі очікують абітурієнти;
- Зелений – абітурієнт викликаний.

Після натискання кнопки наступне вікно програми секретар приймає вигляд (приведені різні випадки: коли в черзі є клієнти, але всі вже знаходяться на прийомі в інших робочих столах – тобто для наступного виклику немає абітурієнтів, коли в черзі є клієнти, абітурієнт ще не підійшов, почалася робота з абітурієнтом).

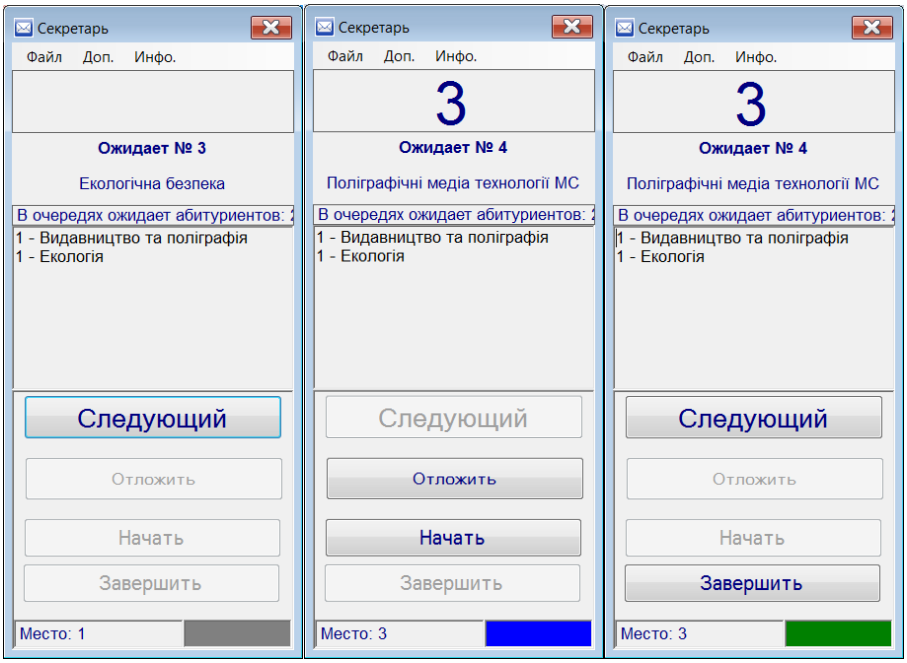

Рис. 4.10.5 – Форма секретаря в різних режимах

Також можна подивитися, скільки в чергах чекають абітурієнтів - пункти меню Інфо – > Показати чергу.

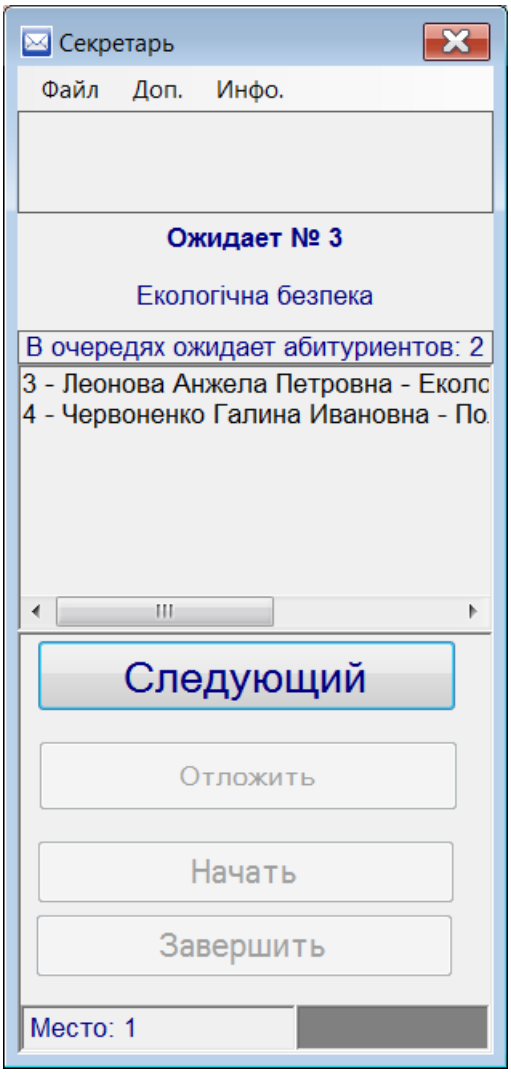

Рис. 4.10.6 – Інформація про чергу

Вихід з програми - пункти меню *Файл>Вихід.*

4.11 Додаток QADisplay.exe

Дана програма служить для того, щоб сповіщати абітурієнтів про рух черги (рис. 4.11.1).

Даний проект складається з наступних модулів:

- Form_Client.cs – головна форма (електронне табло)
- Client.cs class клієнта TcpClient

Після натискання секретарем кнопки «Наступний», запис про цю дію заноситься в базу даних. Дана програма бере усі записи з бази даних і виводить їх на екран. Нові записи виводяться знизу, підсвічуючись сірим коліром. З'явленні записи поступово піднімаються вгору. Тобто більш нові записи витісняють найстарші. Якщо абітурієнт запізнився, то запис виключається з черги і через 10 або 30 хвилин повертається на попереднє місце (після 6 запізнень запис остаточно видаляється з черги).

Якщо людина повністю оформився і секретар натиснув на кнопку «Завершити», а запис про абітурієнта ще знаходиться на екрані, то вона видаляється. Тобто всі абітурієнти з повністю оформленими документами прибираються з екрану. Зроблено з огляду на звільнення місця для нових абітурієнтів (рис. 4.11.2).

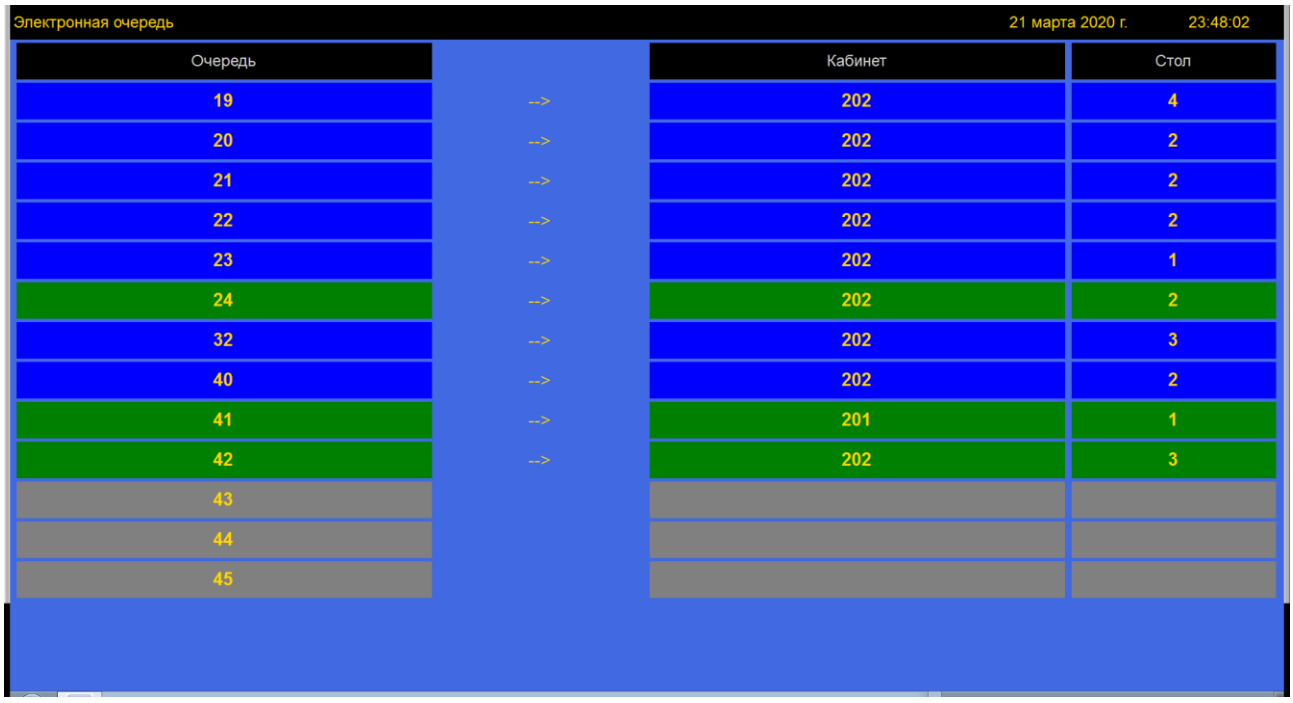

Рис. 4.11.1 – Додаток Qdisplay

Вийти з програми можна викликом контекстного меню з вікна заголовка програми(рис. 46).

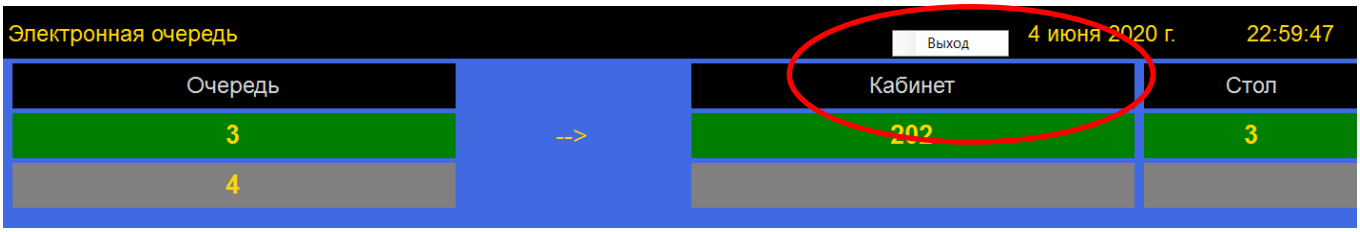

Рис. 4.11.2– Контекстне меню виходу з програми

Параметри великого настінного монітора

1. Колір індикаторів: сірий - для порожнього рядка або рядка з талоном, який ще не викликаний секретарем, світло-синій - для рядків з талоном, який викликаний секретарем, світло-зелений - для рядків з талоном, який знаходиться на обслуговуванні(абітурієнт прибув до робочого столу)
2. Кількість інформаційних рядків на моніторі - 15
3. Кожний інформаційний рядок складається з числа, що позначає номер черги, числа, позначаючого номер кабінету і числа, яке позначає номер робочого місця (наприклад, 133 --> 15 1);

5 КЕРІВНИЦТВО КОРИСТУВАЧА

В якості керівництва користувача пропонуємо файл Instrukcija_Oueue.1.pdf. Даний файл доступний до виклику з усіх розроблених додатків за винятком Qdisplay. Окремі фрагменти керівництва наведені на малюнках 5.1 - 5.3. Повністю керівництво користувача приведено в додатку Б.

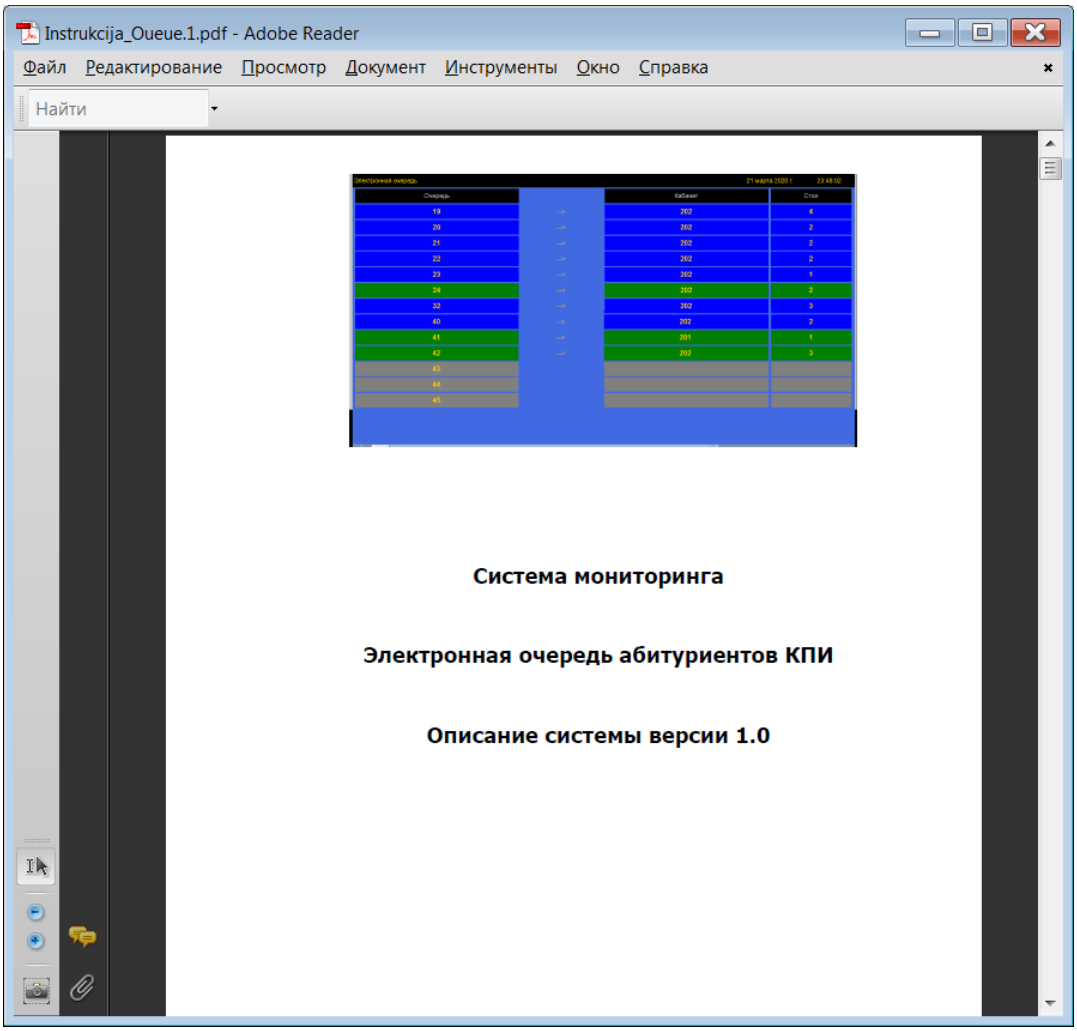

Рис.20 – Керівництво користувача

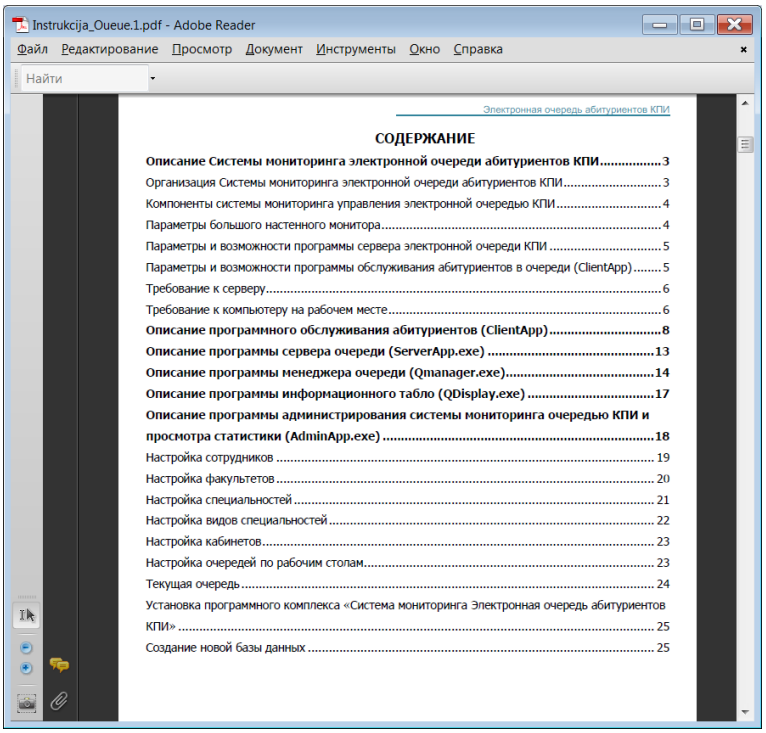

Рис. 5.2 – Керівництво користувача

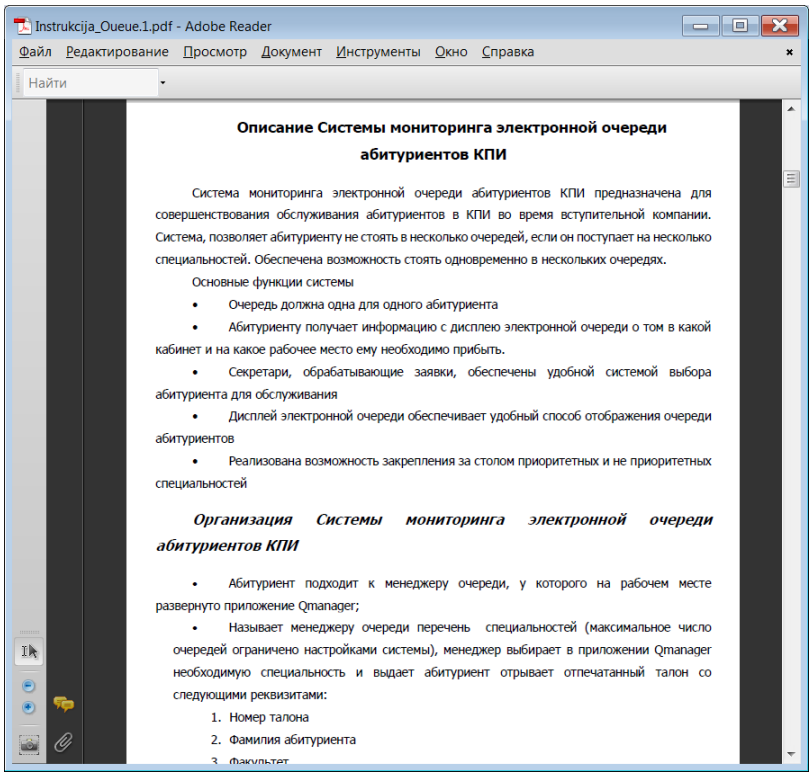

Рис. 5.3 – Керівництво користувача

6 КЕРІВНИЦТВО ПРОГРАМІСТА

6.1 Установка ПЗ

Установка програмного комплексу «Система моніторингу електронної черги вступників КПІ» на сервері включає в себе наступні кроки:

- розпакування саморозпакувального архіву QueueKPI.exe в зазначену папку (рис.50).;
- установка серверу MS SQL і розміщення на ньому бази даних QBase.mdf;
- установка програм сервера і конфігурації «Система моніторингу електрона черга вступників КПІ» ServerApp і AdminApp;
- установка клієнтських робочих місць ClientApp на робочих місцях секретарів;
- установка додатку Qmanager на робочому місці менеджера черг;
- установка додатку QDisplay на ПК для виведення інформації на інформаційне табло;

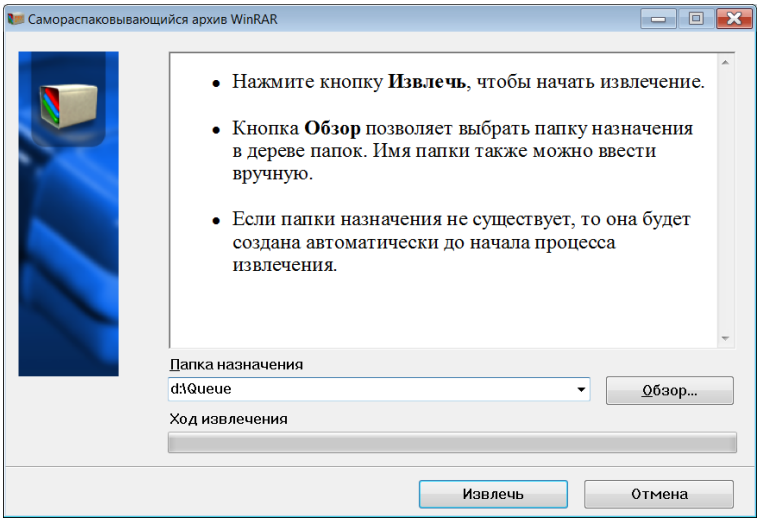

Рис. 6.1– Розпакування інсталяційного архіву

6.2 Створення нової бази даних

Для створення нової (пустої) бази даних необхідно використовувати файл Queue.sql на сервері MS SQL.

6.3 Вимоги до серверу і клієнтських комп’ютерів

- Процесор частотою 1 GHz (або більш потужний);
- Не менш ніж 512 MB вільної оперативної пам’яті (RAM);
- 500 MB вільного простору на жорсткому диску;
- Мережева плата;
- Операційна система Windows 7 або вище;
- Фіксований IP адрес.

ВИСНОВКИ

Об’єктом дослідження даної випускної кваліфікаційної роботи є система моніторингу електронної черги вступників КПІ.

У процесі виконання випускної кваліфікаційної роботи був проведений аналіз організаційної структури КПІ, аналіз структури факультетів, спеціальностей, були визначені функції, що виконуються, вивчена технологія збору, обробки і передачі інформації, зроблений аналіз комплексних технічних і програмних засобів, наявних в організації, а також з’ясований їх взаємозв’язок.

Підсумком випускної кваліфікаційної роботи є розроблена система моніторингу електронної черги вступників КПІ.

У процесі розробки програми були поетапно реалізовані наступні завдання:

1. Проведено аналіз предметної області, вивчена організаційна структура та об'єкти управління;
2. Обґрунтована необхідність використання системи моніторингу електронної черги вступників КПІ;
3. Розроблено формалізований опис об’єкта автоматизованої інформаційної системи;
4. Здійснена розробка інфологічної моделі;
5. Здійснена розробка даталогічної моделі БД;
6. Здійснена розробка фізичної моделі БД;
7. Створена база Qbase.mdf;

В процесі розробки додатку для підключення до бази даних були поетапно реалізовані наступні задачі:

1. В Visual Studio розроблені додатки для роботи з базою даних Qbase.mdf – використані файли AdmiApp.exe і ServerApp.exe;
2. Використані файли для взаємозв’язку з ServerApp.exe: Qdisplay.exe, ClientApp.exe и Qmanager.exe

3. Проведено тестування системи;

Всі поставлені задачі виконані, додаток має зручний інтерфейс і виконує всі необхідні функції, які були визначені при проектуванні.

Перспективи розвитку проекту:

1. Розробка web-інтерфейса для підключення до системи моніторингу електронної черги вступників КПІ;

2. Розробка мобільного додатку для підключення до системи моніторингу електронної черги вступників КПІ.

3. Опрацювання питань формування детальних звітів по роботі системи моніторингу електронної черги вступників КПІ.

В результаті роботи над випускною кваліфікаційною роботою була проведена самостійна дослідницька робота, а також систематизовані, закріплені і поглиблені теоретичні і практичні знання, сформовані на протязі навчання по спеціальності, які були використані для рішення конкретної задачі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Автоматизация проектирования вичислительних систем. Язики, моделирование и бази даних / ред. М. Брейер. - М.: Мир, 2015. – 463 с.

2. Головна сторінка КПІ [Електронний ресурс] — Режим доступу: https://kpi.ua/kpi_about

3. Васюткина И.А. Технология разработки Об’єктно-ориентированних программ на C# в Visual Studio.Net. – Новосибирск: НГТУ, 2015

4. ГОСТ 19.701-90 (ИСО 5807-85) ЕСПД. Схеми алгоритмов,

5. Дейт, К. Дж. Введение в системи баз даних, 8-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2016. – 1328 с.

6. Интернет-университет информационних технологий. Комплекс бесплатних учебних курсов INTUIT.RU (версия 1.0). [Электронний ресурс] – Режим доступа: WWW.URL: http://www.intuit.ru.

7. Исаев, Г. Н. Проектирование информационних систем: учебное пособие – М.: ОМЕГА-Л, 2015. – 424 с.

8. Кузнецов С.Д. Бази даних: учебник для студ. Учреждений висшего проф. Образования / С.Д. Кузнецов. – М. : Издательский центр «Академия», 2012. – 496 с.

9. Ладиженский, Г. Интеграция приложений такая, как она есть [Электронний ресурс] – Режим доступа: WWW.URL: http://www.citcity.ru/ 16663/

10. Марченко А. Л. Основи програмирования на C# 2.0 : учебное пособие- М.: Интернет-Университет Информационних Технологий : Бином , 2007, 551 с. ил.

11. Нортроп, Тони. Основи разработки приложений на платформе Microsoft .NET Framework : экзамен-536 MCTS; [пер. с англ.] - М. : Русская редакция ; СПб. : Питер , 2007 , 842 с. ил.

12. Павловская Т. А. C#. Програмирование на язике високого уровня : [учебник для вузов по направлению "Информатика и вичислительная техника"] - М. [и др.] : Питер , 2007, 2009, 432 с. ил.

13. Петцольд, Чарльз. Програмирование с использованием Microsoft Windows Forms : [пер. с англ.]- М. : Русская редакция ; СПб. : Питер , 2015 , 410 с. ил.

14. Портал «Открытие системи». [Электронний ресурс] – Режим доступа: WWW.URL: http://www.osp.ru/.

15. Сатунина, А.Е. Управление проектом корпоративной информационной системи предприятия/ А.Е. Сатунина, Л.А. Сисоева. – М. : Финанси и статистика; ИНФРА-М, 2009 – 352 с

16. Соловьев, И.В. Проектирование информационних систем. Фундаментальний курс / И.В. Соловьев, А.А. Майоров; Моск. гос. ун-т геодезии и картографии . - М. : Акад. проект, 2015. - 399 с.

17. Фаронов В.В. Програмирование на язике C# : учебний курс - СПб.: Питер, 2007 , 239 с. ил.

18. Хомоненко А.Д., Циганков В.М., Мальцев М.Г. Бази даних: Учебник для висших учебних заведений / Под ред. Проф. А.Д. Хоменко. – 4-е изд., доп.и перераб. – СПб.: КОРОНА принт, 2014, 736 с.

ДОДАТОК 1

Система кодування інформації

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ8130_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_ТМ8130_20Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_ТМ8130_20Б 12-1	Server.cs	Додаток серверу
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_8130_20Б 12-2	Form1.cs	Додаток адміністратора
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_8130_20Б 12-2	Form1.cs	Додаток дисплею
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_8130_20Б 12-2	Form1.cs	Додаток менеджера
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_8130_20Б 12-2	Form1.cs	Додаток секретаря
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕ ПС_8130_20Б 12-2	QBase.mdf	База даних

ДОДАТОК 2

Система кодування інформації

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ8130_20Б

Аркушів 20

Київ – 2020

Лістинг Модуля DB_Connection приложения ServerApp

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
namespace ServerApp
{
    static class DB_Connection
    {
        /// <summary>
        /// Путь к базе даних
        /// </summary>
        public static string dbPath = @"d:\DirQBase\";
        public static  string connectstring;
        public static int countMaxQueue=0;
        /// <summary>
        /// Название бази даних
        /// </summary>
        private static string dbName = "QBase.mdf";
        /// <summary>
        /// База даних приложения
        /// </summary>
        private static SqlConnection con;
        /// <summary>
        /// Текущая хранимая команда бази даних
        /// </summary>
        private static SqlCommand dBcom;
        /// <summary>
        /// Текущий запрос к базе даних
        /// </summary>
        private static string dbQuery;
        /// <summary>
        /// Строка ошибки бази даних
        /// </summary>
        private static string dbError;
        /// <summary>
        /// Подключаемся к базе даних
        /// </summary>
        /// <exception cref="SqlException">Невозможно подключиться к базе даних</exception>
        static public void DBConnect()
        {
            try
            {
                string pathfile = Directory.GetCurrentDirectory() + "\\Path.inf";
                if (File.Exists(pathfile))
                {
                    try
                    {
                        FileStream fs = new FileStream(pathfile, FileMode.Open);
                        StreamReader sr = new StreamReader(fs, Encoding.UTF8);
                        connectstring = sr.ReadLine();
                        sr.Close();
```

```
                    }
                    catch (Exception ex)
                    {
                        connectstring = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" + dbPath +
dbName + @";Integrated Security=True;Connect Timeout=30";
                    }
                }
                else
                {
                    connectstring = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" + dbPath +
dbName + @";Integrated Security=True;Connect Timeout=30";
                }

                dbError = string.Empty;
                string connectionString = connectstring;
                con = new SqlConnection(connectionString);
                dbQuery = "SELECT countMaxQueue  from settingsQueue where idSetting=1";
                dBcom = new SqlCommand(dbQuery, con);
                con.Open();
                countMaxQueue = (Convert.ToInt32(dBcom.ExecuteScalar()));
                con.Close();
            }
            catch (SqlException ex)
            {
                dbError = "Невозможно подключиться к базе даних! " + ex.ToString();
            }
            finally
            {
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }
        }
        /// <summary>
        /// Получить последнюю ошибку бази даних
        /// </summary>
        /// <returns>Получить последнюю ошибку бази даних</returns>
        static public string GetError()
        {
            return dbError;
        }

        /// <summary>
        /// Найти користувача в таблици пользователей
        /// </summary>
        /// <param name="Name">Заданное имя користувача</param>
        /// <param name="Parol">ЗаДаний Password користувача</param>
        /// <returns>Найден ли пользователь в соответствующей таблице</returns>
        /// <exception cref="SqlException">Невозможно считать данние с таблици
пользователей</exception>
        static public bool FindUser(string Name, string Parol)
        {
            try
            {
                dbError = string.Empty;
                dbQuery = "SELECT COUNT(*) FROM [Users] WHERE [Login] = @Login AND [Password] =
@Parol";
                dBcom = new SqlCommand(dbQuery, con);
```

```
                dBcom.Parameters.Add(new SqlParameter { ParameterName = "@Login", DbType =
System.Data.DbType.String, Value = Name });
                dBcom.Parameters.Add(new SqlParameter { ParameterName = "@Parol", DbType =
System.Data.DbType.String, Value = Parol });
                con.Open();
                int count = Convert.ToInt32(dBcom.ExecuteScalar());  con.Close();
               // MessageBox.Show(count.ToString());
                return count == 1 ? true : false;

            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици пользователей! " +
ex.ToString();
                return false;
            }
        }

        static public int GetUserCode(string User)
        {
            try
            {
                dbError = string.Empty;
                dbQuery = "SELECT [IdUser] FROM [Users] WHERE [Login] = @User";
                dBcom = new SqlCommand(dbQuery, con);
                dBcom.Parameters.Add(new SqlParameter { ParameterName = "@User", DbType =
System.Data.DbType.String, Value = User });
                con.Open();
                object a = dBcom.ExecuteScalar(); con.Close();
                return Convert.ToInt32(a);
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици пользователей! " +
ex.ToString();
                return -1;
            }
        }

        //универсальний метод добавления даних в grid
        static  public void gettable(string sql, DataSet ds)
        {
            try
            {
                con.Open();
                SqlDataAdapter da = new SqlDataAdapter(sql, con);
                da.Fill(ds);
                con.Close();
            }
            catch
            {
                MessageBox.Show("Ошибка получения даних з таблици");
            }
            finally
            {
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }
```

```
        }
        static public string GetServerIsRunning(string idManager)
        {
            try
            {

                return "yes";
            }
            catch (SqlException ex)
            {
                dbError = "Не удается подключиться! " + ex.ToString();
              //  MessageBox.Show("3: " + dbError);
                return "no";
            }
         //   MessageBox.Show("4: " + "Mestake");
         //   return "no";
        }
        static public string setintstatus0(string idUser, string Number, string intStatus, string
idTable, string time)
        {
            try
            {
                string sql = "";
                if (intStatus == "1")
                {
                     sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timeCall=@time
where Number=@Number and DATEDIFF(day, timeRegistration, GETDATE()) = 0";
                }
                if (intStatus == "2")
                {
                     sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timeServiceS=@time
where Number=@Number  and DATEDIFF(day, timeRegistration, GETDATE()) = 0;";
                }
                if (intStatus == "11" || intStatus == "12" || intStatus == "13" || intStatus ==
"14" || intStatus == "15" || intStatus == "16" || intStatus == "17")
                {
                    sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timePutOff=@time
where Number=@Number  and DATEDIFF(day, timeRegistration, GETDATE()) = 0;";
                }
                if (intStatus == "9" || intStatus == "10" || intStatus == "17")
                {
                    sql = @"update   QueueCurrent set
                  idUser=@idUser,
intStatus=@intStatus,
idTable=@idTable,
timeEndService=@time
where Number=@Number  and DATEDIFF(day, timeRegistration, GETDATE()) = 0;";
                }
```

```
dBcom = new SqlCommand(sql, con);
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@idUser", DbType =
System.Data.DbType.Int32, Value = Convert.ToInt32(idUser) });
if (intStatus == "17")
    dBcom.Parameters.Add(new SqlParameter { ParameterName = "@intStatus", DbType =
System.Data.DbType.Int32, Value = 10 }); else dBcom.Parameters.Add(new SqlParameter {
ParameterName = "@intStatus", DbType = System.Data.DbType.Int32, Value =
Convert.ToInt32(intStatus) });
if (idTable == "0") dBcom.Parameters.Add(new SqlParameter { ParameterName = "@idTable", DbType =
System.Data.DbType.Int32, Value = DBNull.Value});
else dBcom.Parameters.Add(new SqlParameter { ParameterName = "@idTable", DbType =
System.Data.DbType.Int32, Value = Convert.ToInt32(idTable) });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@time", DbType =
System.Data.DbType.DateTime, Value = Convert.ToDateTime(time) });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@Number", DbType =
System.Data.DbType.Int32, Value = Convert.ToInt32(Number)});
con.Open();
dBcom.ExecuteScalar();
con.Close();
                        return ("1");

            }
            catch (SqlException ex)
            {


                return "Ошибка" + ex.ToString();
            }
     finally
            {
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }

        }
 static public string InsertClientToQueue_GetNamber(string idManager, string idSpecialityS
,string idDetailSpecialityS,
     string timeRegistrationS, string infoClient)
        {
            try
            {
                string[] charArr = infoClient.Split(new Char[] { '|' });
                dbQuery = "SELECT count(*)  from QueueCurrent where nameClient='" +
charArr[0].Trim() + "' and idSubSpeciality=" + idDetailSpecialityS +
                    " and  DATEDIFF(day, timeRegistration, GETDATE()) = 0"; ;
                dBcom = new SqlCommand(dbQuery, con);
                con.Open();
                int a = (Convert.ToInt32(dBcom.ExecuteScalar()));
                con.Close();
                if (a > 0) return "Такой абитуриент на данную специальность уже зарегистрирован";
                dbQuery = "SELECT count(*)  from QueueCurrent where nameClient='" +
charArr[0].Trim() + "'" +
                    " and  DATEDIFF(day, timeRegistration, GETDATE()) = 0";
                dBcom = new SqlCommand(dbQuery, con);
                con.Open();
                a = (Convert.ToInt32(dBcom.ExecuteScalar()));
                con.Close();
```

```
                if (a == countMaxQueue) return "Абитуриент зарегистрирован на макимально
возможное количество очередей: "+countMaxQueue.ToString();
                string sql = @"SELECT * from QueueCurrent where  DATEDIFF(day, timeRegistration,
GETDATE()) = 0";
                DataSet ds= new DataSet();
                gettable(sql, ds);
 int max=1;
                if (ds.Tables[0].Rows.Count !=0)

                {
                    dbQuery = "SELECT max(Number)  from QueueCurrent where   DATEDIFF(day,
timeRegistration, GETDATE()) = 0";
                    dBcom = new SqlCommand(dbQuery, con);

                    con.Open();
                     a = ( Convert.ToInt32(dBcom.ExecuteScalar()));
                     max = a + 1;
                    con.Close();
                }
                     sql = @"INSERT INTO QueueCurrent
                  (Number,
idSpecialty,
timeRegistration, idManager, intStatus, idSubSpeciality,
nameClient,dateBirthClient,FDP,MAN,VUO,OLIMP)
VALUES (" + max.ToString() + "," + idSpecialityS + ",@timereg," + idManager + ",0," +
idDetailSpecialityS + ",@nameClient,@dateBirthClient,@FDP,@MAN,@VUO,@OLIMP)";
dBcom = new SqlCommand(sql, con);
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@timereg", DbType =
System.Data.DbType.DateTime, Value = Convert.ToDateTime(timeRegistrationS) });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@nameClient", DbType =
System.Data.DbType.String, Value = charArr[0].Trim() });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@dateBirthClient", DbType =
System.Data.DbType.Date, Value = Convert.ToDateTime( charArr[1].Trim() )});
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@FDP", DbType =
System.Data.DbType.String, Value = charArr[2].Trim() });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@MAN", DbType =
System.Data.DbType.String, Value = charArr[3].Trim() });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@VUO", DbType =
System.Data.DbType.String, Value = charArr[4].Trim() });
dBcom.Parameters.Add(new SqlParameter { ParameterName = "@OLIMP", DbType =
System.Data.DbType.String, Value = charArr[5].Trim() });
con.Open();
dBcom.ExecuteScalar();
con.Close();

                    {

                        return ((max).ToString());
                    }


            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние ! " + ex.ToString();

                return "Ошибка";
            }
        finally
            {
```

```
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }

        }
        /// <summary>
        /// Получить Login користувача
        /// </summary>
        /// <returns>Login користувача</returns>
        /// <exception cref="SqlException">Невозможно считать данние с таблици
пользователей</exception>
        static public string GetLogin(string code)
        {
            try
            {
                dbError = string.Empty;
                dbQuery = "SELECT [Login] FROM [Users] WHERE [IdUser] = @code";
                dBcom = new SqlCommand(dbQuery, con);
                dBcom.Parameters.Add(new SqlParameter { ParameterName = "@code", DbType =
System.Data.DbType.String, Value = code });
                con.Open(); object a = dBcom.ExecuteScalar(); con.Close();
                return Convert.ToString(a);
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици пользователей! " +
ex.ToString();
                return string.Empty;
            }
        }
        static public string GetFullName(string idUserS)
        {
            try
            {
                dbError = string.Empty;
                dbQuery = "SELECT [FullName] FROM [Users] WHERE [idUser] = " + idUserS;
                dBcom = new SqlCommand(dbQuery, con);
           //     dBcom.Parameters.Add(new SqlParameter { ParameterName = "@login", DbType =
DbType.String, Value = login });
                con.Open(); object a = dBcom.ExecuteScalar(); con.Close();
                return Convert.ToString(a);
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици пользователей! " +
ex.ToString();
                return string.Empty;
            }
        }

        //универсальний метод добавления даних в grid столи
        static public ArrayList getTableWTable()
        {


             try
            {
                dbError = string.Empty;
```

```
                dbQuery = @"SELECT WTable.idTable, WTable.nameTable, Cabinet.nameCabinet
FROM     Cabinet INNER JOIN
                  WTable ON Cabinet.idCabinet = WTable.idCabinet Order by nameCabinet,
nameTable";
                dBcom = new SqlCommand(dbQuery, con);


                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                adapter.Fill(table);
                con.Close();
                ArrayList arr = new ArrayList();
                //формируем Отправленние
                foreach (DataRow dr in table.Rows)
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() + "~"
                        + dr.ItemArray[2].ToString() );
                }
                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Рабочие столи! " +
ex.ToString();
                return null;
            }
        }
        static public ArrayList GetCurrentQueue()
        {
            try
            {
                dbError = string.Empty;
                dbQuery = @"SELECT
idQueueCurrent,
Number,
idTable,
idSpecialty,
intStatus,
idSubSpeciality,
nameClient,
dateBirthClient,
FDP,
MAN,
VUO,
OLIMP,
timePutOff
FROM     QueueCurrent where (intStatus<>9) and (intStatus<>10) and intStatus<>17 and
(DATEDIFF(day, timeRegistration, GETDATE()) = 0) order by Number  ";
                dBcom = new SqlCommand(dbQuery, con);

                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                adapter.Fill(table);
                con.Close();
                ArrayList arr = new ArrayList();

                foreach (DataRow dr in table.Rows)
```

```
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() + "~"
                        + dr.ItemArray[2].ToString() + "~"
                         + dr.ItemArray[3].ToString() + "~"
                        + dr.ItemArray[4].ToString() + "~"
                         + dr.ItemArray[5].ToString() + "~"
                        + dr.ItemArray[6].ToString() + "~"
                         + dr.ItemArray[7].ToString() + "~"
                        + dr.ItemArray[8].ToString() + "~"
                         + dr.ItemArray[9].ToString() + "~"
                        + dr.ItemArray[10].ToString() + "~"
                         + dr.ItemArray[11].ToString() + "~"
                         + dr.ItemArray[12].ToString() );
                  //  s += dr.ItemArray[1].ToString();
                }

                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Черги! " + ex.ToString();
                return null;
            }
        }
        static public ArrayList GetCurrentQueueForDisplay()
        {
            try
            {
                dbError = string.Empty;
                dbQuery = @"SELECT idQueueCurrent, Number, intStatus, idTable,timePutOff
FROM     QueueCurrent where intStatus<>9 and intStatus<>10 and intStatus<>17  and DATEDIFF(day,
timeRegistration, GETDATE()) = 0 order by Number ";

                dBcom = new SqlCommand(dbQuery, con);

                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                adapter.Fill(table);
                con.Close();
                ArrayList arr = new ArrayList();
                //формируем массив
                foreach (DataRow dr in table.Rows)
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() + "~"
                        + dr.ItemArray[2].ToString() + "~"
                         + dr.ItemArray[3].ToString() + "~"
                         + dr.ItemArray[4].ToString());

                }

                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Специальности! " +
ex.ToString();
                return null;
```

```
            }
        }
        static public ArrayList GetSpeciality()
        {
            try
            {
                dbError = string.Empty;
                dbQuery = @"SELECT
Specialty.IdSpecialty,
Specialty.NameSpecialty,
Specialty.KodSpecialty,
Specialty.idFaculty
FROM   Specialty
ORDER BY Specialty.NameSpecialty";
                dBcom = new SqlCommand(dbQuery, con);

                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                adapter.Fill(table);
                con.Close();
                ArrayList arr = new ArrayList();
                //формируем
                foreach (DataRow dr in table.Rows)
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() + "~"
                        + dr.ItemArray[2].ToString() + "~"
                        + dr.ItemArray[3].ToString() );
                }
                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Специальности! " +
ex.ToString();
                return null;
            }
        }
static public ArrayList GetCountQueueCurrent()
        {
            try
            {
                dbError = string.Empty;
                dbQuery = @"SELECT COUNT(QueueCurrent.idQueueCurrent) AS Количество,
Specialty.NameSpecialty
FROM     Specialty INNER JOIN
                  QueueCurrent ON Specialty.IdSpecialty = QueueCurrent.idSpecialty where
QueueCurrent.intStatus<>9 and QueueCurrent.intStatus<>10  and intStatus<>17 and DATEDIFF(day,
QueueCurrent.timeRegistration, GETDATE()) = 0
GROUP BY Specialty.NameSpecialty";
                dBcom = new SqlCommand(dbQuery, con);

                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                adapter.Fill(table);
                con.Close();
                ArrayList arr = new ArrayList();
                //формируем
```

```
                foreach (DataRow dr in table.Rows)
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() );
                }
                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Специальности! " +
ex.ToString();
                return null;
            }
        }
        static public ArrayList GetDetali()
        {
            try
            {
                dbError = string.Empty;
                dbQuery = @"SELECT
idDetailSpecialty,
nameDetailSpecialty,
idSpecialty
FROM     DetailSpecialty
ORDER BY nameDetailSpecialty";
                dBcom = new SqlCommand(dbQuery, con);

                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                con.Close();
                adapter.Fill(table);
                ArrayList arr = new ArrayList();
                //формируем
                foreach (DataRow dr in table.Rows)
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() + "~"
                        + dr.ItemArray[2].ToString() );
                }
                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Специальности! " +
ex.ToString();
                return null;
            }
        }
        static public ArrayList GetFaculties()
        {
            try
            {
                dbError = string.Empty;
                dbQuery = @"SELECT
idFaculty,
NameFaculty
FROM     Faculties
ORDER BY NameFaculty";
                dBcom = new SqlCommand(dbQuery, con);
```

```
                DataTable table = new DataTable();
                con.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(dBcom);
                con.Close();
                adapter.Fill(table);
                ArrayList arr = new ArrayList();
                //формируем
                foreach (DataRow dr in table.Rows)
                {
                    arr.Add(dr.ItemArray[0].ToString() + "~"
                        + dr.ItemArray[1].ToString() );
                }
                return arr;
            }
            catch (SqlException ex)
            {
                dbError = "Не удается прочитать данние с таблици Специальности! " +
ex.ToString();
                return null;
            }
        }



    }
}
```

Лістинг модуля ClassQBase додатку AdmiApp

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Data.OleDb;
using System.Data.SqlClient;
namespace AdminApp
{
    class ClassQBase
    {

        public SqlConnection con = new SqlConnection();
        public string connection;
        SqlDataAdapter daspr;
        DataSet dsspr;

        public string fiouser;
        public int koduser;
        public string loginuser;
        public string roleuser;
        public string doljnuser;
        public string tekparol;
        public int nds;
        //инициализация
        public ClassQBase()
```

```
        {
            string pathfile = Directory.GetCurrentDirectory() + "\\Path.inf";
            if (File.Exists(pathfile))
            {
                try
                {
                    FileStream fs = new FileStream(pathfile, FileMode.Open);
                    StreamReader sr = new StreamReader(fs, Encoding.UTF8);
                    connection = sr.ReadLine();
                   sr.Close();
                }
                catch (Exception ex)
                {
                    connection = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" +
         @"d:\DirQBase\QBase.mdf" +
         ";Integrated Security=True;Connect Timeout=30";
                }
            }
            else
            {
                connection = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" +
         @"d:\DirQBase\QBase.mdf" +
         ";Integrated Security=True;Connect Timeout=30";
            }
            con.ConnectionString = connection;
      //      nds = Convert.ToInt32(getname("select ндс from настройки"));


        }


        //получаем данние по сотруднику
        public void Login( String s, String p, ref Int32 count)
        {

            string sql = @"
                    SELECT
                        Login, Password, Fullname,
 Admin, IdUser FROM Users
                    WHERE  (Login = '" + s + "') AND (Password= '" + p + "') and admin = 1";
            try
            {
                con.Open();
                DataSet ds = new DataSet();
                SqlDataAdapter da = new SqlDataAdapter(sql, con);
                DataTable dt = new DataTable();
                da.Fill(dt);
                con.Close();
                con.Open();

                count = dt.Rows.Count;
                if (count > 0)
                {
                    string passwordDB = dt.Rows[0].ItemArray[1].ToString();//доступ для
конкретной строки
                    if (p == passwordDB.Trim())
                    {
                        count = 1;
                        loginuser = dt.Rows[0].ItemArray[0].ToString();
                        tekparol = dt.Rows[0].ItemArray[1].ToString();
```

```
                    fiouser = dt.Rows[0].ItemArray[2].ToString();
                    roleuser = "admin";
                    koduser = Int32.Parse(dt.Rows[0].ItemArray[4].ToString());
                    doljnuser = "admin";

                }
                else
                {
                    count = 0;
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
        finally
        {
            if (con != null && con.State != ConnectionState.Closed)
            {
                con.Close();
            }
        }
    }
    //универсальний метод добавления даних в grid
    public void gettable(string sql, DataSet ds)
    {
        try
        {
            con.Open();
            SqlDataAdapter da = new SqlDataAdapter(sql, con);
            da.Fill(ds);
            con.Close();
        }
        catch
        {
            MessageBox.Show("Ошибка получения даних з таблици");
        }
     finally
        {
            if (con != null && con.State != ConnectionState.Closed)
            {
                con.Close();
            }
        }
    }
    //универсальний метод добавления даних в grid
    public void getTableUsers(  ref  SqlDataAdapter da)
    {
        string sql = @"SELECT IdUser, FullName, Login, Telefon, Mailadres, Works,admin FROM
Users
";
        con.Open();
        da = new SqlDataAdapter(sql, con);
        con.Close();
      }
     //универсальний метод добавления даних в grid
    public void getTableDetailSpecialty(  ref  SqlDataAdapter da)
    {
```

```
            string sql = @"SELECT DetailSpecialty.idDetailSpecialty,
DetailSpecialty.nameDetailSpecialty, Specialty.NameSpecialty, Faculties.nameFaculty
FROM     DetailSpecialty INNER JOIN
                  Specialty ON DetailSpecialty.idSpecialty = Specialty.IdSpecialty INNER JOIN
                  Faculties ON Specialty.idFaculty = Faculties.idFaculty
ORDER BY Faculties.nameFaculty, Specialty.NameSpecialty";
            con.Open();
            da = new SqlDataAdapter(sql, con);
            con.Close();
          }
        //получаем специальности
        public void getTableSpecialty(  ref  SqlDataAdapter da)
        {
            string sql = @"SELECT Specialty.IdSpecialty,  Specialty.NameSpecialty,
Specialty.KodSpecialty, Faculties.nameFaculty
FROM     Faculties INNER JOIN
                  Specialty ON Faculties.idFaculty = Specialty.idFaculty
ORDER BY Specialty.NameSpecialty";
            con.Open();
            da = new SqlDataAdapter(sql, con);
            con.Close();
          }

        //поиск в таблице
        public void poisk(DataGridView grid,string s)
        {
            if (s.Trim() == "")
            {
                MessageBox.Show("Нет запроса");
                return;
            }
            for (int i = 0; i < grid.RowCount; i++)
            {
                for (int j = 0; j < grid.ColumnCount; j++)
                {
                    grid.Rows[i].Cells[j].Selected = false;
                    if (grid.Rows[i].Cells[j].Value != null)
                        if
(grid.Rows[i].Cells[j].Value.ToString().ToLower().Contains(s.ToLower()))
                        {
                            grid.Rows[i].Cells[j].Selected = true;
                            break;
                        }
                }
            }
        }
        //универсальний метод получения значения
        public  string  getname(string sql)
        {
            con.Open();
            SqlDataAdapter da = new SqlDataAdapter(sql, con);
            DataSet ds = new DataSet();
            da.Fill(ds);
            con.Close();
            string s = "";
            if (ds.Tables[0].Rows.Count==1)
            s = ds.Tables[0].Rows[0][0].ToString();

            return s;
        }
```

```
        //универсальний метод добавления даних в grid
        public void gettableQueue(ref  SqlDataAdapter da)
        {
            string sql = @"SELECT QueueCurrent.idQueueCurrent, QueueCurrent.Number,
DetailSpecialty.nameDetailSpecialty, Specialty.NameSpecialty, Faculties.nameFaculty,
                  QueueCurrent.timeRegistration, QueueCurrent.timeCall,
QueueCurrent.timeServiceS, QueueCurrent.timeEndService, QueueCurrent.timePutOff,
QueueCurrent.timeWaitingS,
                  Users.FullName, QueueCurrent.nameClient, QueueCurrent.FDP, QueueCurrent.MAN,
QueueCurrent.VUO, QueueCurrent.OLIMP, QueueCurrent.idTable
FROM     QueueCurrent INNER JOIN
                  DetailSpecialty ON QueueCurrent.idSubSpeciality =
DetailSpecialty.idDetailSpecialty INNER JOIN
                  Users ON QueueCurrent.idManager = Users.IdUser INNER JOIN
                  Specialty ON DetailSpecialty.idSpecialty = Specialty.IdSpecialty INNER JOIN
                  Faculties ON Specialty.idFaculty = Faculties.idFaculty;
";
            con.Open();
            da = new SqlDataAdapter(sql, con);
            con.Close();
        }


        //универсальний метод добавления даних в grid закази
        public void getTableWTable(ref  SqlDataAdapter da)
        {
            string sql = @"SELECT WTable.idTable, WTable.nameTable, Cabinet.nameCabinet
FROM     Cabinet INNER JOIN
                  WTable ON Cabinet.idCabinet = WTable.idCabinet Order by nameCabinet,
nameTable";
            con.Open();
            da = new SqlDataAdapter(sql, con);
            con.Close();
        }
        //универсальний метод добавления даних в grid закази
        public void gettabletovars(ref  SqlDataAdapter da)
        {
            string sql = @"SELECT Товари.КодТовара, Товари.НаименованиеТовара, Товари.ЦенаБезНДС
FROM Товари
WHERE (((Товари.ЭтоРабота)=0));
";
            con.Open();
            da = new SqlDataAdapter(sql, con);
            con.Close();
        }

        //вставка записи
        public void insertrecord(string nametable, DataGridView grid, Boolean updategrid)
        {
            Program.modified = false;
            Program.insert = true;
            selectrecord(nametable, 0, "");

//обновляем таблицу если изменяли
            if (Program.modified &&  updategrid)
            {
                if (nametable != "specialtyontable")
                {
                    selecttable(nametable, grid);
                    grid.CurrentCell = grid.Rows[grid.RowCount - 1].Cells[1];
```

```
                }
            }
        }
        //вибор записей
        public void selectrecord(string nametable, int id,string name)
        {
            if (Program.readonlybase == true)
            {
                MessageBox.Show("Додаток запущено в режиме чтения. Сервер уже запущен");
                return;
            }
            switch (nametable)
            {
                case "faculties":
                    FormSpr w = new FormSpr("faculties", id, "idFaculty", "nameFaculty", name);
                    w.ShowDialog();
                    break;
                case "cabinet":
                    FormSpr w2 = new FormSpr("Cabinet", id, "idCabinet", "nameCabinet", name);
                    w2.ShowDialog();
                    break;

                case "specialty":
                    fspeciality w4 = new fspeciality(id);
                    w4.ShowDialog();
                    break;

                case "detailspecialty":
                    fdetailSpecialty w6 = new fdetailSpecialty(id);
                    w6.ShowDialog();
                    break;




                case "settingsQueue":
                    FormSpr w111 = new FormSpr("settingsQueue", id, "idSetting", "countMaxQueue",
name);
                    w111.ShowDialog();
                    break;
                case "users":
                    fsotrudnik w12 = new fsotrudnik(id);
                    w12.ShowDialog();
                    break;
                case "wtable":
                    fwtable w13 = new fwtable(id);
                    w13.ShowDialog();
                    break;
                case "specialtyontable":
                    fspecialtyontable w14 = new fspecialtyontable(id);
                    w14.ShowDialog();
                    break;




                default: // все иное
                    MessageBox.Show("Таблица " + nametable + " не найдена");
                    return;
```

```
            } //конец case
        }
        //удаление записей
        public void deleterecord(string nametable, DataGridView grid)
        {
            if (Program.readonlybase == true)
            {
                MessageBox.Show("Додаток запущено в режиме чтения. Сервер уже запущен");
                return;
            }
            try
            {
                int nrows = indexselectedgrid(grid);
                if (nrows == -1)
                {
                    MessageBox.Show("Не вибрана запись");
                    return;
                }
                int index = 0;
                if (!Int32.TryParse(grid[0, nrows].Value.ToString(), out index))
                {
                    MessageBox.Show("Ошибка удаления ");
                    return;
                }
                string pole = grid.Columns[0].HeaderText;
                SqlCommand command1 = new SqlCommand();

                switch (nametable)
                {

                    case "faculties":
                        command1 = new SqlCommand(@" SELECT COUNT(*) from specialty  where
idFaculty = " + index.ToString(), con);
                        break;
                    case "specialty":
                        command1 = new SqlCommand(@" SELECT COUNT(*) from detailSpecialty  where
idSpecialty = " + index.ToString(), con);
                        break;
                    case "detailspecialty":
                        command1 = new SqlCommand(@" SELECT COUNT(*) from detailspecialty  where
idDetailSpecialty = -1" , con);
                        break;
                    case "cabinet":
                        command1 = new SqlCommand(@" SELECT COUNT(*) from WTable  where idCabinet
= " + index.ToString(), con);
                        break;


                    case "specialtyontable":
                        command1 = new SqlCommand(@" SELECT COUNT(*) from specialtyontable  where
idspecialtyontable = -1") ;//+ index.ToString(), con);
                        break;

                    case "Сотрудники":
                        command1 = new SqlCommand(@" SELECT COUNT(*) from Закази  where
(КодМенеджера = " + index.ToString() +
                                                              ") or (КодИсполнителя=" +
index.ToString()+")", con);
                       // MessageBox.Show(command1.CommandText);
                        break;
```

```
                    default:
                        MessageBox.Show("Таблица отсутствует в списке для обработки");
                        return;
                }
                con.Open();
//Откриваем соединение с БД
                object result = command1.ExecuteScalar();
                int a = Convert.ToInt32(result);//Виполняем запрос.
                con.Close();
                if (a > 0)
                {
                    MessageBox.Show("Ошибка удаления, Дана запись используется в других
таблицах.",
                                        "Ошибка удаления даних", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                    return;
                }
                string s=grid[1, nrows].Value.ToString() ;
                if (nametable == "specialtyontable") s = grid[3, nrows].Value.ToString();

                var resultt = MessageBox.Show("Ви действительно хотите удалить " + s+ "?",
"Вибор",
                            MessageBoxButtons.YesNo,
                            MessageBoxIcon.Question);
                // If the no button was pressed ...
                if (resultt == DialogResult.Yes)
                {

                             command1 = new SqlCommand(@"delete FROM " + nametable + " where " +
pole + "=" + index.ToString(), con);


                    con.Open();
                    command1.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Удаление прошло успешно ");
                    grid.Rows.RemoveAt(nrows);
                    Program.modified = true;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка удаления " + ex.ToString(),
                    "Ошибка удаления даних", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
            finally
            {
                if (con != null && con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }
        }
        //проверка вибора строки в таблице
        public int indexselectedgrid(DataGridView grid)
        {
            if (grid.Rows.Count == 0)
```

```
            {
                return -1;
            }
            if (grid.CurrentRow != null)
                return grid.CurrentRow.Index;
            else return -1;
        }
        //редактирование записи
        public void edittable(string nametable, DataGridView grid, Boolean updategrid)
         {
             if (Program.readonlybase == true)
             {
                 MessageBox.Show("Додаток запущено в режиме чтения. Сервер уже запущен");
                 return;
             }
             int nrows = indexselectedgrid(grid);
             if (nrows == -1)
             {
                 MessageBox.Show("Не вибрана строка для редактирования");
                 return;
             }
             int id = Int32.Parse(grid[0, nrows].Value.ToString());
             Program.insert = false;
             Program.modified = false;
             selectrecord(nametable, id, grid[1, nrows].Value.ToString());
             if (Program.modified && updategrid)
             {
                 selecttable(nametable, grid);
                 grid.CurrentCell = grid.Rows[nrows].Cells[1];
             }
         }
        //виборка з таблици
        public    void selecttable(string nametable, DataGridView grid)
        {
            try
            {
                switch (nametable)
                {
                    case "users":
                        getTableUsers(ref  daspr);
                        grid.DataSource = null;
                        dsspr = new DataSet();
                        daspr.Fill(dsspr);
                        grid.DataSource = dsspr.Tables[0];
                        grid.Columns[0].Visible = false;
                        break;
                    case "specialty":
                        getTableSpecialty(ref  daspr);
                        grid.DataSource = null;
                        dsspr = new DataSet();
                        daspr.Fill(dsspr);
                        grid.DataSource = dsspr.Tables[0];
                        grid.Columns[0].Visible = false;
                        break;
                    case "wtable":
                        getTableWTable(ref  daspr);
grid.DataSource = null;
                        dsspr = new DataSet();
                        daspr.Fill(dsspr);
                        grid.DataSource = dsspr.Tables[0];
```

```
grid.Columns[0].Visible = false;
                        break;
                    case "detailspecialty":
                        getTableDetailSpecialty(ref  daspr);
                        grid.DataSource = null;
                        dsspr = new DataSet();
                        daspr.Fill(dsspr);
                        grid.DataSource = dsspr.Tables[0];
                        grid.Columns[0].Visible = false;
                        break;

                    case "QueueCurrent":
                        gettableQueue(ref  daspr);
                        dsspr = new DataSet();
                        daspr.Fill(dsspr);
                        grid.DataSource = dsspr.Tables[0];
                        break;

                    default:
                        string sql = "Select * from " + nametable;  //универсальний метод
добавления даних в grid
                        dsspr = new DataSet();
                        gettable(sql, dsspr);
                        grid.DataSource = dsspr.Tables[0];
                        grid.Columns[0].Visible = false;
                        break;
                }//switch

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }


    }
}
```

ДОДАТОК 3

Система кодування інформації

Опис програмного модулю

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ8130_20Б 13-1

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Результат роботи є програмний комплекс «Система моніторингу електронної черги вступників КПІ», який складається з 5 додатків і бази даних MS SQL, опис процесу створення систему моніторингу електронної черги вступників КПІ, опис розгортання і установки програмного комплексу, керівництва користувача і програміста.

Розробка систему здійснюється в середовищі Microsoft Visual Studio 2019, SQL Server Management.

В процесі роботи досліджувалися програмні продукти і технології, які використовуються для створення систем взаємодії додатків з базами даних MS SQL на основі класів System.Data.SqlClient, який є постачальником даних .NET FrameWork для джерел даних MS SQL Server.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Використання електронних системи управління потоками абітурієнтів в університетах допомагають змінити і підвищити якість обслуговування. У випадку необхідності організовують запис абітурієнтів на прийом по часу і даті. Системи електронних черг дозволяють на основі в процесі роботи даних оптимізувати обслуговування або розробляти нові методики, а також оперативно їх обробляти.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Основні функції розробленої системи:

- черга повинна бути одна для однієї людини;
- система повинна автоматично виводити на загальнодоступному моніторі номер викликаного абітурієнта;
- секретарі, які обробляють заявки, повинні бути забезпечені зручною системою відстеження стану черги і виклику абітурієнта для обслуговування;
- повинен бути доступний інтерфейс відображення поточної черги на загальнодоступному моніторі;
- реалізувати можливість закріплення за робочим місцем секретаря пріоритетних і не пріоритетних спеціальностей.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний принцип роботи додатку такий:

1) користувач запускає сервер;
2) налаштовуємо адміністративну панель до необхідних параметрів;
3) менеджер заносить абітурієнтів до черги;
4) запускається дисплей зі станом черги;
5) секретар обслуговує абітурієнта.

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Серверна частина програмного продукту "Система моніторингу вступників електронної черги вступників КПІ" є додатком, написаним мовою програмування C#. Клієнтський інтерфейс написаний мовою програмування C#.

Для роботи з базою даних використовувалася СУБД MS SQL SERVER

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— база даних структури університету та спеціальностей.

Вихідними даними є:

— додаток для роботи з сервером, додаток для роботи з адміністративною панеллю, додаток для роботи менеджера, додаток дисплей, додаток секретаря